UNIVERSITAT POLITÈCNICA DE CATALUNYA

TESIS DOCTORAL

APORTACIÓN A LA GENERACIÓN DE UMBRALES ADAPTATIVOS A PARTIR DE ENVOLVENTES DE SISTEMAS CON MODELOS APROXIMADOS. APLICACIÓN A LA DETECCIÓN Y DIAGNOSIS ROBUSTA DE FALLOS EN PROCESOS INDUSTRIALES

Autor: VICENÇ PUIG CAYUELA

Director: JOSEBA-JOKIN QUEVEDO CASIN

Capítulo 7:

OPTIMIZACIÓN GLOBAL APROVECHANDO LA ESTRUCTURA DEL PROBLEMA (I): PROGRAMACIÓN GEOMÉTRICA

En este capítulo presentaremos un método de optimización que aprovecha la estructura particular del problema de optimización que aparece al formular el problema de generación de envolventes a partir del nuevo algoritmo basado en optimización y en la ventana deslizante. Recordemos cual es la formulación de dicho problema de optimización para cada una de las variables de estado del sistema en un determinado instante de tiempo $k = n$ en función del valor de los estados en el instante de tiempo discreto $k = n\text{-}L$ donde L es la longitud de la ventana temporal, nos conduce a una problema de optimización cuya función objetivo es:

$$f(A,B,x_{n-L}) = A^{L}x_{n-L} + A^{L-1}Bu_{n-L} + \cdots + Bu_{n-1} \quad (7.1)$$

es una función no lineal de varias variables en concreto de: A, B y x_{n-L}. La no linealidad aparece debido a que la variable A esta elevada a potencias, y debido a los productos entre la variables A y x_{n-L} y las variables A y B. Observando la estructura de dicha función objetivo se observa que es un polinomio en el que intervienen las variables A, B y x_{n-L} multiplicadas entre si y a la vez elevadas a potencias. Dicho polinomio está formado por la suma de monomios de las mismas variables A, B y x_{n-L}. Como veremos a continuación un problema de optimización que reúne estas características recibe el nombre de problema de optimización geométrico. Este tipo de problema de optimización aparece muy a menudo en problemas de ingeniería y ha sido estudiado por muchos investigadores, disponiéndose actualmente de multitud de bibliografía sobre como resolverlo. Para el caso de que los coeficientes de los monomios sean todos ellos positivos existen una transformación logarítmica del problema, de forma que el problema transformado es convexo y por lo tanto tiene una única solución, por lo tanto global. Cuando los coeficientes pueden tomar valores positivos y negativos entonces se deben de aplicar algunas las técnicas de optimización global, que aprovechando la estructura particular del problema, consigan determinar el óptimo global.

7.1 Programación Geométrica

En este apartado consideraremos problemas del tipo:

$$\begin{aligned} &\textit{minimizar} && f(x) \\ &\textit{sujeto a } g_i(x) \le 1 && \textit{para } i=1,\ldots,m \\ & && x>0 \end{aligned} \quad (7.2)$$

donde cada una de las funciones f y g_i es un posinomial de $x \in E_n$ y donde las variables x se suponen que toman valores estrictamente positivos por la propia naturaleza del problema. Un **posinomial** es una función compuesta de términos del tipo

$$T_k = \alpha_k \prod_{j=1}^{n} x_j^{a_{kj}} \quad (7.3)$$

donde $\alpha_k > 0$, y donde los exponentes a_{kj}, $j=1,\ldots,n$, son números racionales que pueden ser de cualquier signo. En particular, para $x > 0$, tenemos que $T_k > 0$ también. Por lo tanto, las funciones objetivo y restricción se pueden escribir, respectivamente, como:

$$f(x) = \sum_{k \in J_o} T_k \qquad (7.4)$$

y

$$\sum_{k \in J_o} g_i(x) = \sum_{k \in J_i} T_k \quad \text{para } i = 1, \ldots, m \qquad (7.5)$$

donde la colección de conjuntos índice $J_0, J_1, \ldots, J_m$ son mútuamente disjuntos, y donde

$$J_0 \cup J_1 \cup \ldots \cup J_m \equiv \{1,2,\ldots,M\} \qquad (7.6)$$

representa un total de M términos, cada uno de ellos del tipo $T_k = \alpha_k \prod_{j=1}^{n} x_j^{a_{kj}}$. Los problemas geométricos (GP) de este tipo son *problemas de programación posinomial*. Cuando los coeficientes α_k pueden ser negativos, entonces las funciones f(x) y $g_i(x)$ se denominan signomiales, y el problema GP se conoce como *problema de programación signomial*. En cualquier caso, el problema se denomina *problema de programación geométrica*.

Los problemas de programación geométrica aparecen frecuentemente en aplicaciones de ingeniería donde las variables de decisión x son variables de diseño que precisan tomar sólo valores positivos para que tengan sentido, y donde las funciones objetivo y restricción modelan relaciones físicas fundamentales que por naturaleza son del tipo posinomial, o bien, se pueden transformar en dicho tipo de funciones. Existe una transformación que simplifica considerablemente este tipo de problema, convirtiéndolo en un sistema de ecuaciones lineales, o bien, en un problema con restricciones. Esta transformación consta de dos pasos.

La programación geométrica es hoy en día un tema clásico de cualquier libro de programación no lineal. Un buen ejemplo es el capítulo que le dedica Bazaraa [Bazaraa93] en su libro "Nonlinear Programming". Existe una excelente revisión del tema de programación geométrica, sus métodos y aplicaciones realizada por Ecker [Ecker80].

7.2 Programación Posinomial

Tal como hemos visto en el apartado anterior un **posinomio** es una función g de un vector de variables $x \in R^m$ de la forma

$$g(x) = \sum_{i=1}^{N} u_i(x) \qquad (7.7)$$

con

$$u_i(x) = c_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_m^{a_{im}} \qquad i = 1,2,\cdots,N \qquad (7.8)$$

donde los exponentes a_{ij} son números reales arbitrarios y los coeficientes c_i son positivos.

Un problema de **programación posinomial** es un problema de optimización que obedece a la siguiente forma

$$\begin{aligned} (P) \quad & \min g_0(x) \\ & \text{s.t.} \quad g_k(x) \le 1 \text{ para } \quad k = 1,2,\cdots,p \\ & \qquad x > 0 \end{aligned} \qquad (7.9)$$

donde cada $g_k(x)$ para k=0,1,2,...,p es un posinomio.

A continuación presentamos un ejemplo de un problema de este tipo propuesto por Duffin, Peterson y Zener [Duffin67]. Este ejemplo ilustra tanto la notación utilizada para describir un problema de optimización de este tipo como el papel que juega la desigualdad de la media aritmética/geométrica en el

desarrollo de la teoría de la programación geométrica. Así mismo, utilizaremos este ejemplo para motivar un problema de optimización relacionado con el problema de minimización (P), al que denominaremos problema dual.

- **Ejemplo de problema de programación posinomial**

Supongamos que se desea construir un recipiente que permite transportar 400 m^3 de líquido en diversos viajes. Las caras laterales y el fondo deben realizare de un material del que se dispone de una superficie máxima de 4 m^2, mientras que las caras frontal y trasera de deben realizar de un material cuyo valor es de 20 dólares por m^2.
El problema consiste en determinar el mínimo coste en dólares que representa transportar los 400 m^3 de líquido utilizando el recipiente anterior y que cada viaje representa un coste de 10 centavos.

Si utilizamos las variables x_1, x_2 y x_3 para representar la longitud, anchura y altura de la caja, entonces el coste total en dólares vendrá dado por

$$g_o(x) = 40x_1^{-1}x_2^{-1}x_3^{-1} + 40x_2x_3$$

donde el primer término representa el coste del transporte mientras que el segundo término representa el coste del material necesario para construir las caras frontal y trasera. Puesto que el área total de las caras y fondo no puede ser mayor de 4 m^2 (disponibilidad del material), entonces al problema de optimización se le debe añadir la siguiente restricción

$$2x_1x_3 + x_1x_2 \leq 4$$

llegándose al siguiente problema de programación posinomial:

$$\begin{aligned} \min\ & g_o(x) = 40x_1^{-1}x_2^{-2}x_3^{-1} + 40x_2x_3 \\ \text{s.t.}\ & g_1(x) = \tfrac{1}{2}x_1x_3 + \tfrac{1}{4}x_1x_2 \leq 1 \\ & x_j > 0 \text{ para } j = 1,2,3 \end{aligned}$$

Este ejemplo posee cuatro términos posinomiales $u_i(x)$ para $i = 1,2,3,4$. Es conveniente particionar estos índices en bloques de índices consecutivos de forma que se puedan identificar los índices asociados con la función objetivo $g_o(x)$ y los índices asociados con cada función restricción $g_k(x)$. En general, estableceremos que [k], al que denominaremos bloque k, represente los índices asociados con el posinomio $g_k(x)$. En nuestro ejemplo

$$[0] = \{1,2\} \quad \text{y} \quad [1] = \{3,4\}$$

Una matriz importante asociado con cada problema de programación posinomial (P) es la matriz exponente A. En general, A es n × m, donde n es el número de términos posinomiales $u_i(x)$ y m es el número de variables x_j. La i-ésima fila de A contiene los exponentes del i-ésimo término posinomial. En nuestro ejemplo dicha matriz vale:

$$A = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

En general, los términos posinomiales $u_i(x)$ no serán funciones convexas. Por ejemplo, $u(x) = x^{\frac{1}{2}}$ no es convexa sobre el dominio x>0. Como ya sabemos, cuando se minimiza una función no convexa, aparece el problema de los mínimos locales. Sin embargo, este problema no aparecerá para problemas de

programación posinomial (P), puesto que todo óptimo local será también un óptimo global. La razón de esta importante propiedad es que cualquier problema de programación posinomial (P) es equivalente a un problema convexo (donde se minimiza una función convexa en un dominio convexo). Esta equivalencia se establece mediante el siguiente cambio de variable:

$$x_j = e^{z_j} \quad \text{para } j = 1,2,\cdots,m \qquad (7.10)$$

El problema de programación posinomial (P) es equivalente, por lo tanto, al siguiente problema convexo

$$\begin{aligned} &\min \sum_{i\in[0]} c_i e^{A_i Z} \\ &\text{s.t.} \sum_{i\in[k]} c_i e^{A_i Z} \le 1 \quad \text{para } k = 1,2,\ldots,p \end{aligned} \qquad (7.12)$$

donde A_i es la i-ésima fila de la matriz de exponentes A y c_i es el coeficiente positivo asociado con el i-ésimo término posinomial. Se puede comprobar fácilmente que $e^{A_i Z}$ define una función convexa de Z. De donde de deduce que $\log \sum c_i e^{A_i Z}$ define también una función convexa. Por lo tanto, la forma logarítmica del problema anterior es también un problema convexo.

El hecho de que un problema de programación posinomial sea equivalente a un problema convexo conlleva varias consecuencias importantes. La más importante es que la teoría general de programación convexa se puede aplicar al caso posinomial (en su forma equivalente logarítmica) y el problema dual (que veremos a continuación) se puede deducir a través de dicha teoría general. Otra consecuencia importante de dicha equivalencia es que los métodos computacionales basados en los posinomios condensados (que analizaremos a continuación) se puede estudiar en el contexto de la linealización de funciones convexas.

Otra formulación equivalente del problema de programación posinomial (P) se puede obtener aplicando el siguiente cambio de variable sobre el problema convexo equivalente

$$t = Az \qquad (7.13)$$

llegándose al siguiente problema de optimización

$$\begin{aligned} &\min \sum_{i\in[0]} c_i e^{t_i} \\ &\text{s.t.} \sum_{i\in[k]} c_i e^{t_i} \le 1 \quad \text{para } k = 1,2,\ldots,p \\ &t = Az \end{aligned} \qquad (7.14)$$

Las restricciones lineales de dicho problema simplemente requieren que el vector t se encuentre en el espacio de columnas de A. Además, debe observarse que las funciones implicadas son completamente **separables**. La teoría generalizada de programación geométrica es una teoría para estudiar una gran clase de problemas separables, siendo el problema equivalente que acabamos de determinar un ejemplo.

- **El papel de la desigualdad entre las medias aritmética-geométrica**

Duffin, Peterson y Zener [Duffin67] utilizan una generalización de la desigualdad existente entre las medias aritmética y geométrica para obtener las cotas inferiores del mínimo del problema de programación posinomial (P). Esta desigualdad generalizada establece que si U y δ son vectores en R^N con U>0 y $\delta \ge 0$, entonces

$$\sum_{i=1}^{N} U_i^{\lambda} \ge \prod_{i=1}^{N} \left(\frac{U_i}{\delta_i} \right)^{\delta_i} \lambda^{\lambda} \qquad (7.15)$$

donde $\lambda = \sum_{i=1}^{N} \delta_i$.

En particular, para cualquier k, si consideramos los términos posinomiales $u_i(x)$ para $i \in [k]$, de forma que $g_k(x) = \sum_{i \in [k]} u_i(x)$, entonces la desigualdad generalizada anterior se transforma para este caso en

$$\sum_{i \in [k]} u_i(x)^{\lambda_k} \geq \prod_{i \in [k]} \left(\frac{u_i(x)}{\delta_i} \right)^{\lambda_k} \lambda_k^{\lambda_k} \qquad (7.16)$$

donde $\lambda_k = \sum_{[k]} \delta_i$. Esta desigualdad es válida para k = 0,1,...,p donde p es el número de restricciones del problema de programación posinomial (P).

Para k = 0, supongamos que escogemos δ_i para $i \in [0]$, de forma que $\lambda_0 = \sum_{i \in [0]} \delta_i = 1$. Entonces, la desigualdad anterior se traduce para este caso en

$$g_o(x) \geq \prod_{i \in [0]} \left(\frac{u_i(x)}{\delta_i} \right)^{\delta_i} \qquad (7.17)$$

Si x es factible para el problema de programación posinomial (P), entonces debe satisfacer cada una de las restricciones $g_k(x) \leq 1$ para k = 1,2,...,p, de forma que para cada una de ellas la desigualdad generalizada se transforma en

$$1 \geq g_k(x)^{\lambda_k} \prod_{i \in [k]} \left(\frac{u_i(x)}{\delta_i} \right)^{\delta_i} \lambda_k^{\lambda_k} \quad \text{para } k = 1,2,\cdots,p \qquad (7.18)$$

Multiplicando la desigualdad obtenida para k=0 por las p desigualdades anteriores se obtiene

$$g_o(x) \geq \prod_{i=1}^{n} \left(\frac{u_i(x)}{\delta_i} \right)^{\delta_i} \prod_{k=1}^{p} \lambda_k^{\lambda_k} \qquad (7.19)$$

donde n es el número total de términos posinomiales del problema de programación posinomial (P). En el ejemplo, que hemos presentado al principio de este apartado, la parte derecha de la desigualdad que acabamos de obtener se transforma en

$$\left(\frac{40 x_1^{-1} x_2^{-1} x_3^{-1}}{\delta_1} \right)^{\delta_1} \left(\frac{40 x_2 x_3}{\delta_2} \right)^{\delta_2} \left(\frac{\frac{1}{2} x_1 x_3}{\delta_3} \right)^{\delta_3} \left(\frac{\frac{1}{4} x_1 x_2}{\delta_4} \right)^{\delta_4} (\delta_3 + \delta_4)^{\delta_3 + \delta_4}$$

Obsérvese que en la expresión anterior la suma de exponentes de la variables x_j es simplemente el producto escalar de la columna j-ésima de la matriz exponente A con el vector δ. Por lo tanto, dicha expresión se puede hacer independiente de x simplemente escogiendo δ de forma que $A^T\delta = 0$.

Por lo tanto, finalmente se obtiene la siguiente desigualdad:

$$g_o(x) \geq \prod_{i=1}^{n} \left(\frac{c_i}{\delta_i} \right)^{\delta_i} \prod_{k=1}^{p} \lambda_k^{\lambda_k} \qquad (7.20)$$

suponiendo que x es factible para el problema de programación posinomial (P) y que:

$$\delta \geq 0$$
$$\sum_{i\in[0]} \delta_i = 1 \qquad (7.21)$$
$$A^T\delta = 0$$

Esta desigualdad es una herramienta muy útil para resolver el problema de programación posinomial (P). Para cualquier δ que satisfaga las condiciones anteriores, la desigualdad que acabamos de encontrar nos proporciona una cota inferior del mínimo del problema de programación posinomial (P). Por lo tanto, dado un vector x factible para el problema (P), se puede encontrar un vector δ que satisfaga las condiciones enunciadas de forma que se cumpla la desigualdad para $g_o(x)$, entonces se puede afirmar que x es el valor óptimo del problema (P). Así mismo, la desigualdad que acabamos de determinar para $g_o(x)$ es una pieza clave en la motivación de la dualidad del problema de programación posinomial.

- **El dual de un problema de programación posinomial**

En el punto anterior, se ha utilizado la relación de desigualdad existente entre las medias aritmética y geométrica, para determinar la mayor de las cotas inferiores del mínimo del problema de programación posinomial (P). Es natural, que a la vista de la desigualdad sobre $g_o(x)$ a la que se ha llegado, se considere el siguiente problema de optimización, al que denominaremos **problema de programación posinomial dual (D)**

$$(D) \quad \max v(\delta) = \prod_{i=1}^{n}\left(\frac{c_i}{\delta_i}\right)^{\delta_i} \prod_{k=1}^{p} \lambda_k^{\lambda_k}$$
$$\text{s.t.} \quad \sum_{i\in[0]} \delta_i = 1 \qquad (7.22)$$
$$A^T\delta = 0$$
$$\delta \geq 0$$

donde la matriz de exponentes A y el vector de coeficientes c se definen de la misma forma que el problema primal (P).

La función objetivo dual v es continua sobre toda la región de factibilidad, aunque, no es diferenciable en puntos de dicha región donde alguna de las componentes de δ sean igual a cero. A pesar de la no diferenciabilidad de la función objetivo dual v, el problema dual (D) posee características interesantes desde el punto de vista computacional.

Así, por ejemplo, las restricciones son lineales y $V(\delta) = \log v(\delta)$ es una función cóncava. Estas propiedades serán aprovechadas por los métodos computacionales que presentaremos en el próximo apartado para resolver el problema de programación posinomial a partir de su formulación dual.

Dado un problema de programación posinomial en forma primal (P) con una región de factibilidad no vacía, en el proceso de solución, algunos términos $u_k(x)$ se pueden acercar a cero. Cuando ocurre esto, el programa primal (P) se denomina degenerado. Si esto mismo le sucede a cada uno de los términos de la función objetivo $g_o(x)$, entonces el programa primal (P) se denomina totalmente degenerado. Cuando un programa primal (P) no es degenerado se denomina canónico.

Existe una importante relación entre los programas primal y dual que permite clasificar los programas primales como canónicos o degenerados a partir de la matriz de exponentes A y de las variables duales δ. Dicha relación establece que, el programa primal (P) es canónico si y sólo si existe un valor de δ^* tal que $A^T\delta^* = 0$ y $\delta^* > 0$. Obsérvese que si (P) es canónica, entonces su dual (D) es factible. Mientras que si un programa primal (P) no es canónico ni totalmente degenerado, entonces eliminando los términos $u_i(x)$ para los cuales $\delta_i = 0$ para todo δ que satisfaga $A^T\delta = 0$ y $\delta \geq 0$, se obtiene un programa reducido que es canónico. Dicho programa reducido es equivalente al programa original (P) en el sentido de que si x^* es el óptimo del programa reducido lo es también del programa original.

En nuestro caso, nos limitaremos a estudiar problemas de programación posinomial canónicos.

- **Relaciones duales**

A continuación, vamos a resumir los aspectos más importantes de la teoría de la dualidad aplicada a problemas de programación posinomial canónicos.

Teorema 1

Si el programa primal (P) es canónico y su región de factibilidad no está vacía, entonces existe un punto x^* donde se alcanza el mínimo global del problema (P).

Teorema 2

Si el programa primal (P) es canónico y existe un punto $\overline{x}$ para el cual $g_k(\overline{x}) < 1$ para k=1,2,...,p entonces:
(i) el programa dual (D) tiene un punto solución δ^* donde se alcanza el máximo absoluto.
(ii) el valor del máximo $v(\delta^*)$ de (D) es igual al valor del mínimo del programa primal (P)
(iii) todo punto x que minimiza el programa primal (P) cumple que:

$$u_i(x) = \begin{cases} \delta_i^* v(\delta^*) & \text{para } i \in [0] \\ \dfrac{\delta_i^*}{\lambda_k(\delta^*)} & \text{para todo } i \in [k] \text{ con } \lambda_k(\delta^*) > 0 \end{cases} \qquad (7.23)$$

La parte (iii) del Teorema 2 proporciona un método de obtener el punto donde se alcanza el mínimo del problema primal (P) suponiendo que el vector dual óptimo tiene el número suficiente de componentes positivos. Para un valor de δ^* dado, el sistema de ecuaciones no lineal anterior es equivalente a un sistema de ecuaciones lineal cuyas variables son log x_j. Dicho sistema lineal determina un punto donde se alcanza el mínimo del problema primal (P) suponiendo que se dispone de suficientes de variables duales positivas de forma que el rango del sistema sea igual al de la matriz de exponentes A.

Una restricción primal $g_k(x) \leq 1$ se denomina inactiva si su eliminación no permite obtener un valor del mínimo inferior. Se puede afirmar que si $g_k(x^*) < 1$ en el punto donde se alcanza el mínimo x^*, entonces dicha restricción es inactiva. Existe una importante relación entre las restricciones inactivas primales y el valor de las variables duales óptimas: una restricción primal $g_k(x) \leq 1$ es inactiva si y sólo si $\lambda_k(\delta) = 0$ para cada vector óptimo dual δ. Por lo tanto, si existe un número suficiente de restricciones primales inactivas, las ecuaciones duales del punto (iii) del Teorema 2 no determinan de forma única el punto donde se alcanza el mínimo del problema primal (P). Ello conlleva, que deben resolverse uno o más problemas adicionales para determinar el punto donde se alcanza dicho mínimo.

Para ilustrar la utilización del problema dual (D) en la solución del problema primal vamos a continuar el ejemplo utilizado a lo largo de este apartado. El problema dual (D) del problema primal presentado en dicho ejemplo es el siguiente:

$$\min \left(\frac{40}{\delta_1}\right)^{\delta_1}\left(\frac{40}{\delta_2}\right)^{\delta_2}\left(\frac{1}{2\delta_3}\right)^{\delta_3}\left(\frac{1}{4\delta_4}\right)^{\delta_4}\left(\delta_3+\delta_4\right)^{\delta_3+\delta_4}$$

$$\text{s.t.} \quad A^T\delta = 0$$

$$\delta_1 + \delta_2 = 1$$

$$\delta \geq 0$$

donde A es la matriz de exponentes del problema primal. Este problema posee un único vector dual factible: $\delta^* = (\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})^T$. Entonces, δ^* es el punto donde se alcanza el óptimo siendo su valor $v(\delta^*)=60$.

En general, si A es de rango completo y hay n variables duales δ_i y m variables primales x_j, entonces $d = n - m - 1$ es la dimensión del conjunto de todas las soluciones a $A^T\delta = 0$ y $\sum_{i\in[0]} \delta_i = 1$. Por lo tanto, el problema dual podría ser reformulado de forma que sólo tuviera d variables. En la literatura, d se conoce como **grado de dificultad** del problema. La experiencia computacional ha demostrado, sin embargo, que algunos problemas con un gran grado de dificultad d pueden ser más fáciles de resolver que aquéllos que poseen un d menor.

Continuando con el ejemplo, a partir del punto (iii) del Teorema 2, todo punto x solución del problema primal del ejemplo debe satisfacer el siguiente sistema de ecuaciones:

$$40x_1^{-1}x_2^{-1}x_3^{-1} = \tfrac{2}{3}60$$

$$40x_2x_3 = \tfrac{1}{3}60$$

$$\tfrac{1}{2}x_1x_3 = \frac{1/3}{2/3}$$

$$\tfrac{1}{4}x_1x_2 = \frac{1/3}{2/3}$$

Este sistema es equivalente a un sistema lineal con 4 ecuaciones con 3 incógnitas log x_j con j=1,2,3 que posee una única solución $x^* = (2,1,\frac{1}{2})^T$ que constituye el punto donde el problema primal (P) alcanza su mínimo.

En general, cuando el grado de dificultad d > 0, se requiere de la utilización de un método iterativo para obtener un punto dual óptimo. Las ecuaciones duales pueden conducir a un conjunto de ecuaciones inconsistente, debido a que el vector δ^* está muy cerca pero no es exactamente igual al verdadero punto óptimo. Así, por ejemplo, se puede realizar una estimación lineal cuadrática del punto x^*.

Las ecuaciones duales proporcionan la siguiente interpretación sobre las variables duales δ_i para $i\in[0]$ asociados con los términos de la función objetivo primal: si δ^* es el óptimo del problema dual, entonces δ^*_i para $i\in[0]$ nos proporciona la fracción del valor del mínimo debido al i-ésimo término de la función objetivo primal $g_o(x)$.

7.3 Programación Signomial

Hasta ahora hemos considerado el caso para el cual cada uno de los términos posinomiales $u_i(x) = c_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_m^{a_{im}}$ tenía un coeficiente c_i positivo. En este apartado, vamos a relajar esta condición y a considerar alguna de las consecuencias que supone dicha relajación. Muchos programas que aparecen de aplicaciones prácticas importantes dejan de ser programas posinomiales debido a que algunos coeficientes c_i son negativos. Tales programas se denominan **programas signomiales** y fueron estudiados en primer lugar por Passy y Wilde [Passy67] y Blau y Wilde [Blau69].

Duffin y Peterson [Duffin73] demostraron que cualquier programa signomial se puede transformar en un programa equivalente de la forma

$$\begin{aligned} &\min g_o(x) \\ &\text{s.t. } g_k(x) \le 1 \quad \text{para } k = 1,2,\cdots,p \\ &\qquad g_k(x) \ge 1 \quad \text{para } k = p+1,\cdots,q \end{aligned} \tag{7.24}$$

donde $g_k(x)$ es un posinomio para k =1,2,...q. Dicho programa equivalente se denomina **programa posinomial inverso**. Para ver como se puede llegar a dicha reformulación, consideremos una restricción de la forma

$$h_1(x) - h_2(x) \leq 1 \qquad (7.25)$$

donde cada $h_i(x)$ es un posinomio. Debe observarse que x satisface dicha restricción si y sólo si existe una variable simple s tal que (x,s) satisfacen

$$h_1(x) \leq s \leq h_2(x) + 1 \qquad (7.26)$$

desigualdad que equivale a las dos restricciones siguientes

$$\begin{aligned} s^{-1} h_1(x) &\leq 1 \\ s^{-1} h_2(x) + s^{-1} &\geq 1 \end{aligned} \qquad (7.27)$$

que son consistentes que la formulación dada para el programa posinomial inverso.

Mientras que los programas posinomiales admiten una reformulación equivalente como programas convexos, examinando la formulación de los programas signomiales como posinomiales inversos se llega a la conclusión que dicha reformulación no es posible. Las restricciones inversas $g_k(x) \geq 1$ definen una región factible no convexa. En general, por lo tanto, los mínimo locales para los problemas signomiales no son mínimos globales.

7.4 Métodos Computacionales para la Programación Signomial

Un problema de programación signomial se puede expresar como:

$$\left.\begin{aligned} &\min f_0(x) = g_o(x) - h_o(x) \\ &\text{s.t. } f_k(x) = g_k(x) - h_k(x) \leq b_k \quad (k = 1, \cdots, q) \\ &\qquad x > 0 \end{aligned}\right\} \text{Poblema P} \qquad (7.28)$$

donde cada función $g_k(x)$ y $h_k(x)$ son **posinomios** que se puede escribir como:

$$g_k(x) = \sum_{i \in IG(k)} c_i \prod_{j=1}^{m} x_j^{\alpha_{ij}} \qquad (7.29)$$

$$h_k(x) = \sum_{i \in IH(k)} c_i \prod_{j=1}^{m} x_j^{\alpha_{ij}} \quad (k = 0,1,\cdots,q) \qquad (7.30)$$

siendo los exponentes α_{ij} números reales arbitrarios y todos los coeficientes c_i positivos. Los números reales b_k (k=1,..., q) puede ser positivos o negativos. Los conjuntos índice IG(k) y IH(k) (k=0,1,...,q) son subconjuntos mútuamente excluyentes de valores enteros consecutivos:

$$\{1,2,\cdots,n\} = \bigcup_{k=0}^{q} [IG(k) \cup IH(k)] \qquad (7.31)$$

y están ordenados de que forma que:

$$i_o < i_o' < i_1 < i_1' < \cdots < i_q < i_q' \qquad (7.32)$$

siempre cumpliéndose que:

$$\begin{aligned} i_k &\in IG(k) \\ i_k' &\in IH(k) \end{aligned} \qquad (7.33)$$

Alguno de los conjuntos IG(k) o IH(k) pueden estar vacíos, aunque los conjuntos:

$$IG(k) \cup IH(k) \quad (7.34)$$

no deben ser vacíos para cualquier k = 0,1, ..., q.

A continuación definiremos los conjuntos IG y IH de la siguiente manera:

$$IG = \bigcup_{k=0}^{q} IG(h)$$
$$IH = \bigcup_{k=0}^{q} IH(k) \quad (7.35)$$

- **Aplicación sobre un ejemplo**

Consideremos el siguiente problema de programación signomial:

$$\left.\begin{aligned} &\min f_0(x) = x_1^2 + 2x_2^2 - x_1 x_2 \\ &\text{s.t. } f_1(x) = x_1 x_2 \le 1 \\ &\quad f_2(x) = 9x_1^{-1} - 4x_1^{-1} x_2 \le 2 \\ &\quad 0.1 \le x_1 \le 10 \\ &\quad 0.1 \le x_2 \le 10 \end{aligned}\right\} \text{Poblema P}$$

Los posinomios contenidos en este problema son los siguientes:

Posinomio	Conjunto Índice	Coeficientes	Exponentes
$g_0(x) = \sum_{i \in IG(0)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = x_1^2 + 2x_2^2$	IG(0) = {1,2}	$c_1 = 1$ $c_2 = 2$	$\alpha_{11} = 2 \quad \alpha_{12} = 0$ $\alpha_{21} = 0 \quad \alpha_{22} = 2$
$g_1(x) = \sum_{i \in IG(1)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = x_1 x_2$	IG(1) = {4}	$c_4 = 1$	$\alpha_{41} = 1 \quad \alpha_{42} = 1$
$g_2(x) = \sum_{i \in IG(2)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = 9x_1^{-1}$	IG(2) = {5}	$c_5 = 9$	$\alpha_{51} = -1 \quad \alpha_{52} = 0$
$h_0(x) = \sum_{i \in IH(0)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = x_1 x_2$	IH(0) = {3}	$c_3 = 1$	$\alpha_{31} = 1 \quad \alpha_{32} = 1$
$h_1(x) = \sum_{i \in IH(1)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = 0$	IH(1) = Ø		
$h_2(x) = \sum_{i \in IH(2)} c_i \prod_{j=1}^{2} x_j^{\alpha_{ij}} = 4x_1^{-1} x_2$	IH(2) = {6}	$c_6 = 4$	$\alpha_{61} = -1 \quad \alpha_{62} = 1$

Para cada posinomio se ha identificado su conjunto índice, sus coeficientes y sus exponentes.

- **Algoritmo de Falk** [Falk73]

El algoritmo de Falk permite obtener la solución global de un problema signomial. El algoritmo funciona siempre que se cumplan las siguientes suposiciones:

1ª Suposición:

"El Problema P tiene al menos una solución global finita x^*"

Sea:

$$v^* = g_o(x^*) - h_o(x^*) \qquad (7.36)$$

Usando la transformación:

$$x_j = e^{z_j} \qquad (7.37)$$

el problema P se transforma en:

$$\left.\begin{aligned} &\min F_0(z) = G_o(z) - H_o(z) \\ &\text{s.t. } F_k(z) = G_k(z) - H_k(z) \le b_k \quad (k = 1, \cdots, q) \end{aligned}\right\} \text{Poblema Q} \qquad (7.38)$$

donde:

$$G_k(z) = \sum_{i \in IG(k)} c_i e^{(\alpha^i, z)} \qquad (7.39)$$

$$H_k(z) = \sum_{i \in IH(k)} c_i e^{(\alpha^i, z)} \quad (k = 0,1, \cdots, q) \qquad (7.40)$$

donde (α^i, z) enlas expresiones de $G_k(z)$ y $H_k(z)$ representa el producto escalar de los vectores:

$$\alpha^i = (\alpha_{i1}, \cdots, \alpha_{im})^T \text{ y } z = (z_1, \cdots, z_m) \qquad (7.41)$$

Debe observarse que las funciones $G_k(z)$ y $H_k(z)$ son funciones convexas puesto que se trata de sumas de funciones convexas. Si todas las funciones $H_k(z)$ (k=0,1, ..., 1) fueran cero, el Problema Q sería un programa convexo (este caso particular se denomina **programación posinomial** su resolución ya se ha tratado en el apartado anterior).

El problema Q se podría continuar transformando hasta conseguir un problema separable definiendo las variables $t_i = (z, \alpha^i)$ y añadiendo estas nuevas definiciones de variables como restricciones del problema Q. El problema resultante alcanzaría una forma muy similar al requerido por el algoritmo de Falk-Soland, faltando sólo determinar las cotas inferiores y superiores de dichas variables. En la variante del algoritmo de Falk-Soland que se presenta, no se utiliza esta última transformación, manipulando de forma implícita estas nuevas variables sin introducirlas de forma explícita.

2ª Suposición:

"Los vectores z_{max} y z_{min} se pueden determinar de forma que el conjunto:

$$R = \{z : z_{min} \le z \le z_{max}\} \qquad (7.42)$$

contenga al menos una solución global z^* del problema Q."

Normalmente estas cotas se pueden determinar a partir de las características del problema, o se pueden fijar artificialmente a valores razonables que definan un conjunto R que contenga todos los posibles puntos de interés. La eficiencia del algoritmo aumenta de forma considerable cuanto más estrechas sean dichas cotas.

Dado el conjunto R, definimos el conjunto relacionado S^1:

$$S^1 = \{z : m_i^1 \le (\alpha^i, z) \le M_i^1 ; i \in IH\} \qquad (7.43)$$

donde:

$$m_i^1 = \min_z \{(\alpha^i, z) : z \in R\} \qquad (7.44)$$

$$M_i^1 = \max_z \{(\alpha^i, z) : z \in R\} \qquad (7.45)$$

para $i \in IH$.

Estas cotas son fácilmente calculables si tenemos en cuenta que:

$$m_i^1 = \sum_{j=1}^{m} \min\left\{\alpha_{ij} z_j^{min}, \alpha_{ij} z_j^{max}\right\} \qquad (7.46)$$

$$M_i^1 = \sum_{j=1}^{m} \max\left\{\alpha_{ij} z_j^{min}, \alpha_{ij} z_j^{max}\right\} \qquad (7.47)$$

para $i \in IH$.

El conjunto S^1 contiene al conjunto R y por lo tanto contiene una solución global de Q. En general, trataremos con conjuntos del tipo:

$$S^t = \left\{z : m_i^t \le \left(\alpha^i, z\right) \le M_i^t \; ; i \in IH\right\} \qquad (7.48)$$

donde $S^t \subseteq S^1$.

El algoritmo generará una secuencia de problemas Q^t, cada uno de los cuales está asociado con un nodo en un árbol de "branch and bound". Los nuevos problemas se crean seleccionando un nodo ya existente Q^t y construyendo nuevos problemas que aproximen mejor el problema Q que Q^t. A cada paso sólo se construyen dos problemas. En el paso 1, sólo tenemos el problema Q^1, en el paso 2 tenemos los problemas Q^1, Q^2 y Q^3, y en general en paso n tenemos los problemas Q^1, ..., Q^{2n-1}.

La forma general del problema Q^t es

$$\left.\begin{array}{l} \min F_o^t(z) = G_o(z) - L_o^t(z) \\ \text{s.t. } F_k(z) = G_k(z) - L_k(z) \le b_k \quad (k = 1, \cdots, q) \\ z \in R \end{array}\right\} \text{Poblema } Q^t \qquad (7.49)$$

donde

$$L_k^t(z) = \sum_{i \in IH(k)} \ell_i^t(z) \qquad (7.50)$$

$$\ell_i^t(z) = \frac{c_i}{M_i^t - m_i^t}\left[\left(M_i^t e^{m_i^t} - m_i^t e^{M_i^t}\right) + \left(e^{M_i^t} - e^{m_i^t}\right)\left(\alpha^i, z\right)\right] \qquad (7.51)$$

Debe observarse que $\ell_i^t(z)$ sobrestima el valor de $c_i e^{(\alpha^i, z)}$ cuando $m_i^t \le (\alpha^i, z) \le M_i^t$, por lo tanto $F_k^t(z)$ subestima $F_k(z)$ para $z \in S^t$, tal como se muestra en la Fig. 7.1. Y que además el problema Q^t es convexo.

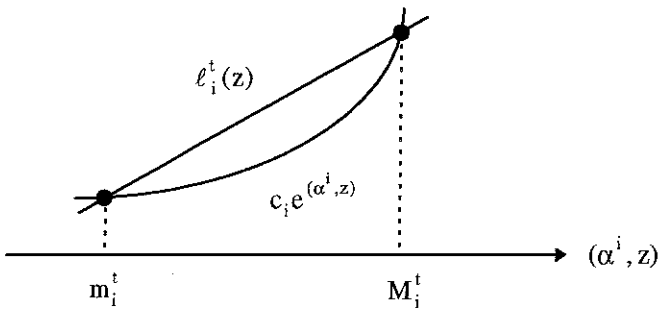


Fig. 7.1 Sobrestimación de $c_i e^{(\alpha^i, z)}$

Sea z^t la solución al problema Q^t si existe, entonces:

$$v^t = \begin{cases} F_o^t(z^t) & \text{si } z^t \text{ existe} \\ +\infty & \text{si } z^t \text{ no existe} \end{cases} \qquad (7.52)$$

$$V^t = \begin{cases} F_o(z^t) & \text{si } z^t \text{ existe} \\ +\infty & \text{si } z^t \text{ no existe} \end{cases} \qquad (7.53)$$

Un nodo (problema) Q^t se denomina intermedio en el paso n-ésimo si no genera ningún subproblema a partir de él.

- **Etapas del Algoritmo de Falk-Soland**

Paso 1. Resolución del problema Q^1 a partir del cual se determinará: z^1, v^1, V^1, $v_b(1)$ y $V_b(1)$. Pasamos al paso 2 con n = 1.

Paso 2. Comprobación de la solución obtenida:

- si $v_b(n) = V_b(n)$, una solución global del problema Q es $z^* = z^t$ donde $V^t = V_b(n)$.
- si $v_b(n) < V_b(n)$ pasamos al paso 3.

Paso 3. Elección un problema Q^t a partir del cual se generarán subproblemas:
se escogerá el subproblema $t \in I(n)$ que cumpla $v^t = v_b(n)$. Pasamos al paso 4.

Paso 4. Elección de un término de ramificación (α^I, z) :

(a) Elección de un $K \in \{0,1,2,...,q\}$ que cumpla que: $L_k^t(z^t) - H_k(z^t)$ sea máximo.

(b) Elección de un $I \in IH(k)$ tal que: $\ell_i^t(z^t) - c_i e^{(\alpha^i, z^t)}$ sea máximo. Pasamos al paso 5.

Paso 5. Creación del paso n+1 (ver Fig. 7.2)

(a) Creación del problema Q^{2n}: se escoge $M_I^{2n} = (\alpha^I, z^t)$ con el I escogido en el paso 4. El resto de valores m_i^{2n} y M_i^{2n} toman el valor correspondiente a m_i^t y M_i^t. Se actualiza $F_k^{2t}(n)$ a partir de:

$$L_k^t(z) = \sum_{i \in IH(k)} \ell_i^t(z)$$

$$\ell_i^t(z) = \frac{c_i}{M_i^t - m_i^t}\left[(M_i^t e^{m_i^t} - m_i^t e^{M_i^t}) + (e^{M_i^t} - e^{m_i^t})(\alpha^i, z)\right]$$

(b) Creación del problema Q^{2n+1}: de la misma forma que el problema Q^{2n} pero cambiando sólo el término $m_I^{2n+1} = (\alpha^I, z^t)$.

(c) Resolución de los problemas Q^{2n} y Q^{2n+1}. Actualización de los valores de $v_b(n)$ y $V_b(n)$. Retorno al paso 2 con n = n + 1.

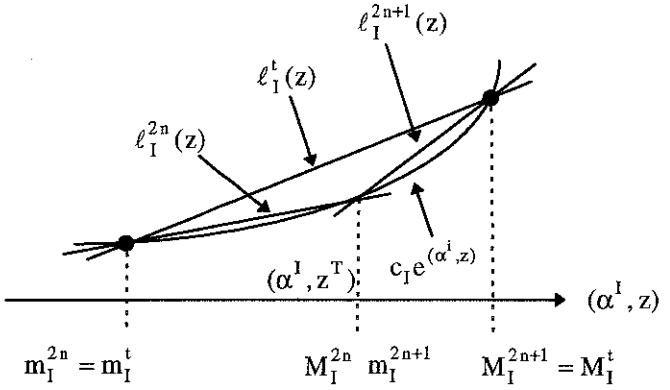


Fig. 7.2 Actualización de la estimación de $c_i e^{(\alpha^i, z)}$ en el paso 5 del algoritmo de Falk

- **Aplicación del algoritmo de Falk sobre un ejemplo**

A continuación aplicaremos el algoritmo que acabamos de describir sobre el ejemplo que habíamos empezado a estudiar anteriormente:

$$\left.\begin{aligned} &\min f_0(x) = x_1^2 + 2x_2^2 - x_1 x_2 \\ &\text{s.t. } f_1(x) = x_1 x_2 \le 1 \\ &\quad f_2(x) = 9x_1^{-1} - 4x_1^{-1} x_2 \le 2 \\ &\quad 0.1 \le x_1 \le 10 \\ &\quad 0.1 \le x_2 \le 10 \end{aligned}\right\} \text{Poblema P}$$

Este problema tiene dos soluciones locales en (1/2, 2) y (4,1/4), con valores de la función objetivo asociados de 7.25 y 15.0125 respectivamente. El óptimo global corresponde a la primera de las soluciones. La forma de la función objetivo dentro de la región factible se puede observar en la Fig. 7.3. mientras la región factible con las soluciones locales identificadas se muestran en la Fig. 7.4

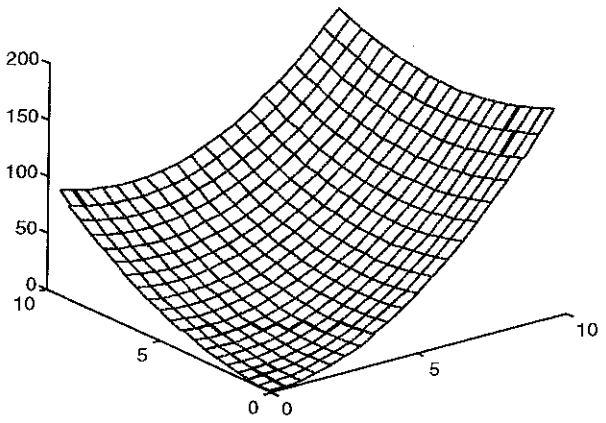


Fig. 7.3 Función objetivo a optimizar

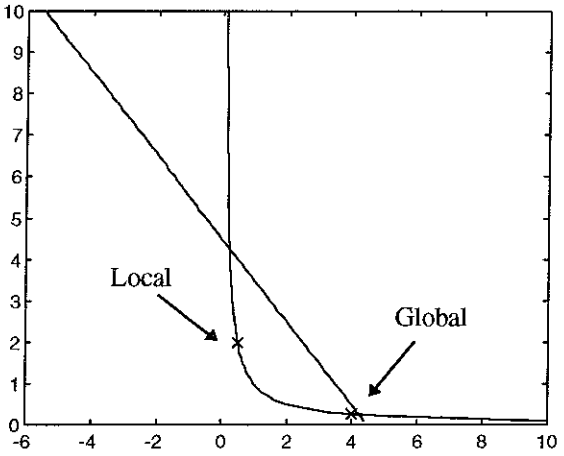


Fig. 7.4 Región factible y soluciones del problema P

Utilizando la transformación: $x_j = e^{z_j}$ el problema P se traduce en el siguiente problema Q:

$$\left.\begin{array}{l} \min F_0(z) = e^{2z_1} + 2e^{2z_2} - e^{z_1+z_2} \\ \text{s.t. } F_1(z) = e^{z_1+z_2} \le 1 \\ \quad F_2(z) = 4.5e^{-z_1} - 2e^{-z_1+z_2} \le 1 \\ \text{R: } \begin{cases} \ln 0.1 \le z_1 \le \ln 10 \\ \ln 0.1 \le z_2 \le \ln 10 \end{cases} \end{array}\right\} \text{Poblema Q}$$

El conjunto S^1 se obtiene a partir del conjunto R de la siguiente manera:

$$\ln 0.01 \le z_1 + z_2 \le \ln 100$$

En la Fig. 7.5 se muestra la región factible para el problema Q.

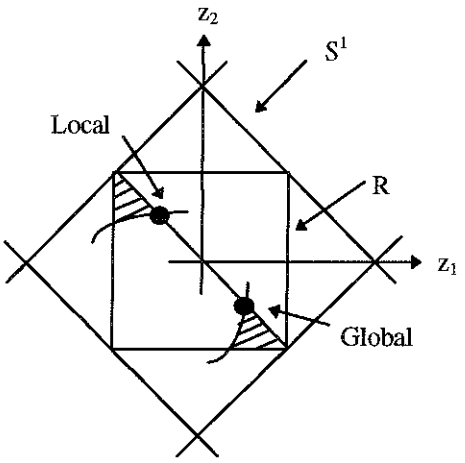


Fig. 7.5 Región factible y soluciones del problema Q

Obsérvese que existen dos funciones cóncavas: $H_0(z) = -e^{z_1+z_2}$ y $H_2(z) = -2e^{-z_1+z_2}$.

Utilizando la notación de la secciones anteriores,

IG(0)={1,2}	IH(0)={3}
IG(0)={1,2}	IH(1) = ϕ
IG(0)={1,2}	IH(2)={6}

por lo tanto, IH = {3,6}. Es decir, sólo existirán dos términos de bifurcación:

$$(\alpha^3, z) = z_1 + z_2$$
$$(\alpha^6, z) = -z_1 + z_2$$

El primer subproblema Q^1 se construye por sobrestimación de la función $e^{z_1+z_2}$ en el intervalo [ln 0.01, ln100] mediante la función

$$\ell_3^1(z) = 10.08563(z_1 + z_2) + 50.00500$$

y de la función $2e^{-z_1+z_2}$ en el intervalo [ln 0.01, ln100] mediante la función

$$\ell_6^1(z) = 21.71233(-z_1 + z_2) + 100.00918$$

por lo tanto, tendremos que

$$\left.\begin{array}{l} \min F_0^1(z) = e^{2z_1} + 2e^{2z_2} - \ell_3^1(z) \\ \text{s.t. } F_1^1(z) = e^{z_1+z_2} \le 1 \\ \quad F_2^1(z) = 4.5e^{-z_1} - \ell_6^1(z) \le 1 \\ \quad \text{R: } \begin{cases} \ln 0.1 \le z_1 \le \ln 10 \\ \ln 0.1 \le z_2 \le \ln 10 \end{cases} \end{array}\right\} \text{Poblema } Q^1$$

La solución z^1 de este problema de optimización convexo se puede calcular con un optimizador clásico basado en técnicas de descenso como MATLAB, obteniéndose

$$z^1 = (0.17328, -0.17328)$$

$$v^1 = -47.17706$$

Puesto que z^1 no es un punto factible del problema original Q, no nos proporciona una cota superior del óptimo de dicho problema . Sin embargo, utilizando z^1 como semilla y aplicando MATLAB directamente al problema original Q nos proporciona la solución factible local (1.38629, -1.38629) y por lo tanto una cota superior de la función objetivo V^1 = 15.12500. La utilización de está técnica para obtener cotas superiores suele ser muy útil, siendo la utilizada en este caso.

Puesto que $v^1 < V^1$, no se cumple la condición de finalización, debiéndose elegir de entre de los subproblemas de las ramificaciones obtenidas. En este caso como sólo existe un subproblema escogemos Q^1. La selección del término de ramificación consiste en seleccionar de entre los términos de ramificación (términos 3 ó 6), aquel cuya diferencia entre el valor del término aproximado y el término exacto sea mayor, o sea, para este caso la mayor de las diferencias calculada para $z = z^1$ de entre

$$\ell_3^1(z^1) - e^{z_1^1+z_2^1} \text{ y } \ell_6^1(z^1) - 2e^{-z_1^1+z_2^1}$$

En este caso, la mayor se produce para el término 6. Por lo tanto, el problema Q^2 es idéntico al problema Q^1 excepto que ℓ_6^2 se define como una sobrestimación lineal más ajustada de la función $2e^{-z_1^1+z_2^1}$ sobre el intervalo [ln0.01, -0.34656] obtenida de acuerdo con la fórmula

$$\ell_i^t(z) = \frac{c_i}{M_i^t - m_i^t}\left[(M_i^t e^{m_i^t} - m_i^t e^{M_i^t}) + (e^{M_i^t} - e^{m_i^t})(\alpha^i, z)\right]$$

Análogamente el subproblema Q^2 es idéntico al problema Q^1 excepto que ℓ_6^3 se define como una sobrestimación más ajustada de la función $2e^{-z_1^1+z_2^1}$ sobre el intervalo [-0.34656, ln 100] obtenida mediante la fórmula anterior.

La solución del problema original Q se llevó a cabo a través de 23 nodos utilizando MATLAB para resolver cada subproblema y aplicando el algoritmo de branch and bound a mano. La convergencia de las cotas inferiores a v^* fue extremadamente lenta, con $v_b(12)$=1.845013. Sin embargo, utilizando MATLAB para obtener cotas superiores válidas intentando resolver el problema Q directamente utilizando como punto semilla cada una de las soluciones de los subproblemas generados, dichas cotas convergieron a la solución global correcta en la tercera iteración: $V_b(3)$ = 7.25000.

7.5 Aplicación de la Programación Signomial a la Generación de Envolventes

La aplicación de la optimización global basada en la formulación del problema como un problema de programación signomial, que se ha presentado en este capítulo, a la resolución del problema de optimización planteado al utilizar el nuevo algoritmo de generación de envolventes que se propone en esta tesis pasa por:

Paso 1. Inicialización. Determinar la expresión de la función objetivo a optimizar para cada uno de los estados del sistema del cual se quieren obtener las envolventes. Esta tarea se puede realizar con cualquier programa de manipulación simbólica como MAPLE.
Paso 2. Formulación del problema de forma que el optimizador signomial basado en algoritmo de Falk pueda resolver el problema de optimización.
Paso 3. Solución del problema de optimización asociado a cada una de las variables de estado utilizando el optimizador signomial basado en el algoritmo de Falk.
Paso 4. Lectura de los resultados de la optimización y generación de las envolventes en el instante de tiempo actual.
Paso 5. Generación de los nuevos problemas de optimización global asociados a cada una de las variables de estado, para obtener el valor de las envolventes en el siguiente instante de tiempo, actualizando las cotas de los estados al inicio de la ventana temporal, a partir de los resultados obtenidos en la solución de los problemas.
Paso 6. Salto al paso 3.

Gráficamente, esta secuencia de pasos se puede representar de la siguiente manera:

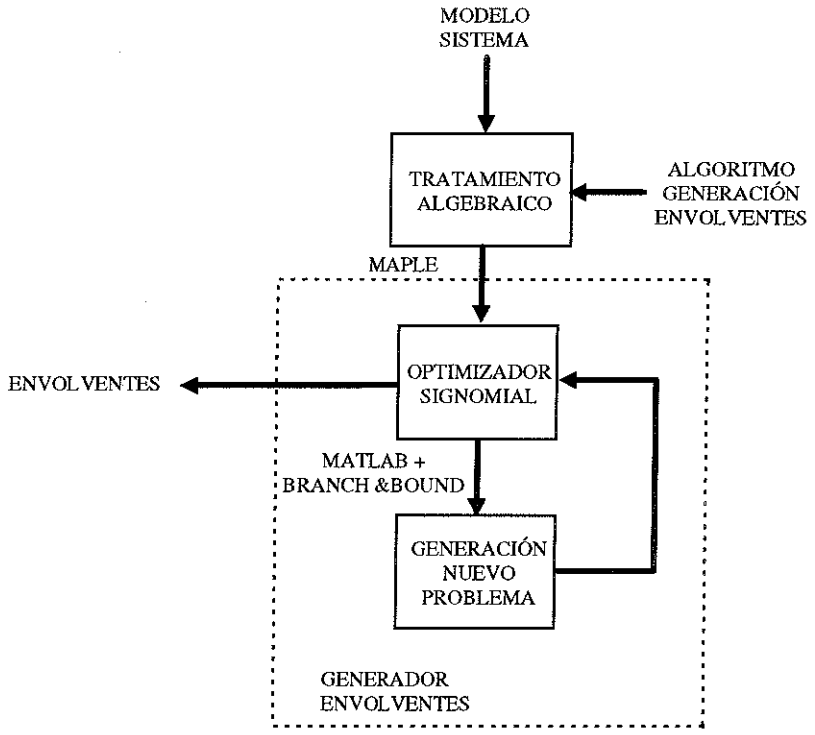


Fig. 7.6 Generación de las envolventes utilizando un optimizador signomial

En esta tesis se han probado el optimizador signomial basado el algoritmo de Falk. Las características principales del cual se han presentado anteriormente se han presentado anteriormente. El optimizador signomial se construido utilizando MATLAB con su correspondiente Toolbox de Optimización.

- **Generación de Envolventes utilizando el algoritmo de Falk**

Para probar las técnicas de optimización global presentadas en este capítulo para resolver el problema de optimización que aparece al generar las envolventes de un sistema utilizando el nuevo algoritmo de generación de envolventes presentado en esta tesis utilizaremos un sistema de segundo orden cuyo modelo en el espacio de estado discreto es el que se presenta a continuación

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

con la siguiente incertidumbre asociada a los parámetros del modelo:

$$a_1 \in [1.3938, 1.4338]$$

$$a_2 \in [0.5865, 0.6265]$$

Utilizando el algoritmo de generación de envolventes presentado en esta tesis, sobre este ejemplo con una vetaba de longitud L = 5, los problema de optimización global que deben resolverse son los siguientes:

Problema 1: "Envolventes para la variable de estado x_1"

Función Objetivo:

$$x_{1k} = f_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^5 - 4x_{1(k-L)}a_1^3a_2 + 3x_{1(k-L)}a_1a_2^2 - a_2x_{2(k-L)}a_1^4 + 3a_2^2x_{2(k-L)}a_1^2 - a_2^3x_{2(k-L)} + a_1^4 - 3a_1^2a_2 + a_2^2 + a_1^3 - 2a_1a_2 + a_1^2 - a_2 + a_1 + 1$$

Restricciones:

$$x_{1(k-L)} = [x^-_{1(k-L)}, x^+_{1(k-L)}]$$

$$x_{2(k-L)} = [x^-_{2(k-L)}, x^+_{2(k-L)}]$$

$$a_1 = [1.3938, 1.4338]$$

$$a_2 = [0.5865, 0,6265].$$

Problema 2: "Envolventes para la variable de estado x_2"

Función Objetivo:

$$x_{2k} = f_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^4 - 3x_{1(k-L)}a_1^2a_2 + x_{1(k-L)}a_2^2 - a_1^3a_2x_{2(k-L)} + 2a_1a_2^2x_{2(k-L)} + a_1^3 - 2a_1a_2 + a_1^2 - a_2 + a_1 + 1$$

Restricciones:

$x_{1(k-L)} = [x^{-}_{1(k-L)}, x^{+}_{1(k-L)}]$

$x_{2(k-L)} = [x^{-}_{2(k-L)}, x^{+}_{2(k-L)}]$

$a_1 = [1.3938, 1.4338]$

$a_2 = [0.5865, 0,6265]$.

Cada uno de estos dos problemas de optimización global se puede catalogar como un problema de optimización signomial, puesto que cada una de las dos funciones objetivo puede descomponerse en la diferencia de dos funciones que a su vez están compuestas por la suma de posinomiales:

Función Objetivo Problema asociado a la variable x_1:

$$x_{1k} = f_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = g_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) - h_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2)$$

donde:

$$g_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^5 + 3x_{1(k-L)}a_1a_2^2 + 3a_2^2x_{2(k-L)}a_1^2 + a_1^4 + a_2^2 + a_1^3 + a_1^2 + a_1 + 1$$

$$h_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = a_2x_{2(k-L)}a_1^4 + 4x_{1(k-L)}a_1^3a_2 + a_2^3x_{2(k-L)} + 3a_1^2a_2 + 2a_1a_2 + a_2$$

Función Objetivo Problema asociado a la variable x_2:

$$x_{2k} = f_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = g_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) - h_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2)$$

donde:

$$g_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^4 + x_{1(k-L)}a_2^2 + 2a_1a_2^2x_{2(k-L)} + a_1^3 + a_1^2 + a_1 + 1$$

$$h_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = 3x_{1(k-L)}a_1^2a_2 + a_1^3a_2x_{2(k-L)} + 2a_1a_2 + a_2$$

Utilizando los siguiente cambios de variable

$$t_1 = x_{1(k-L)}$$

$$t_2 = x_{2(k-L)}$$

$$t_3 = a_1$$

$$t_4 = a_2$$

ambas funciones objetivo se pueden reescribir como:

Función Objetivo Problema asociado a la variable x_1:

$$f_1(t_1,t_2,t_3,t_4) = g_1(t_1,t_2,t_3,t_4) - h_1(t_1,t_2,t_3,t_4)$$

donde:

$$g_1(t_1,t_2,t_3,t_4) = t_1t_3^5 + 3t_1t_3t_4^2 + t_4^2 + t_3^4 + t_3^3 + t_3^2 + t_3 + 1$$

$$h_1(t_1,t_2,t_3,t_4) = t_2t_3^4t_4 + 4t_1t_3^3t_4 + t_2t_4^3 + 3t_3^2t_4 + 2t_3t_4 + t_4$$

Función Objetivo Problema asociado a la variable x_2:

$$f_2(t_1,t_2,t_3,t_4) = g_2(t_1,t_2,t_3,t_4) - h_2(t_1,t_2,t_3,t_4)$$

donde:

$$g_2(t_1,t_2,t_3,t_4) = t_1t_3^4 + t_1t_4^2 + 2t_2t_3t_4^2 + t_3^3 + t_3^2 + t_3 + 1$$

$$h_2(t_1,t_2,t_3,t_4) = 3t_1t_3^2t_4 + t_2t_3^3t_4 + 2t_3t_4 + t_4$$

Si a ambos problemas de optimización les aplicamos el cambio de variable:

$$t_i = e^{z_i}$$

entonces los problemas de optimización se transformarán en:

Problema de Optimización asociado a la variable x_1:

$$\min \quad f_1(z_1,z_2,z_3,z_4) = g_1(z_1,z_2,z_3,z_4) - h_1(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln \bar{x}_{1(k-L)}, \ln \bar{x}_{1(k-L)}]$$

$$z_2 \in [\ln \bar{x}_{2(k-L)}, \ln \bar{x}_{2(k-L)}]$$

$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$

$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4) = e^{z_1+5z_3} + 3e^{z_1+z_3+2z_4} + 3e^{z_2+2z_3+2z_4} + e^{2z_4} + e^{4z_3} + e^{3z_3} + e^{2z_3} + e^{z_3} + 1$$

$$h_1(z_1,z_2,z_3,z_4) = e^{z_2+4z_3+z_4} + 4e^{z_1+3z_3+z_4} + e^{z_2+3z_4} + 3e^{2z_3+z_4} + 2e^{z_3+z_4} + e^{z_4}$$

Problema de Optimización asociado a la variable x_2:

$$\min f_2(z_1,z_2,z_3,z_4) = g_2(z_1,z_2,z_3,z_4) - h_2(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln \bar{x}_{1(k-L)}, \ln \bar{x}_{1(k-L)}]$$

$$z_2 \in [\ln \bar{x}_{2(k-L)}, \ln \bar{x}_{2(k-L)}]$$

$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$

$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_2(z_1,z_2,z_3,z_4) = e^{z_1+4z_3} + e^{z_1+2z_4} + 2e^{z_2+z_3+2z_4} + e^{3z_3} + e^{2z_3} + e^{z_3} + 1$$

$$h_2(z_1,z_2,z_3,z_4) = 3e^{z_1+2z_3+z_4} + e^{z_2+3z_3+z_4} + 2e^{z_3+z_4} + e^{z_4}$$

Centrándonos en el problema de optimización 1, se observa que los posinomios contenidos en este problema son los siguientes:

$$g_o(t) = \sum_{i \in IG(0)} c_i \prod_{j=1}^{4} t_j^{\alpha_{ij}} = t_1 t_3^5 + 3t_1 t_3 t_4^2 + 3t_2 t_3^2 t_4^2 + t_4^2 + t_3^4 + t_3^3 + t_3^2 + t_3 + 1$$

$$h_o(t) = \sum_{i \in IH(0)} c_i \prod_{i=1}^{4} t_j^{\alpha_{ij}} = t_2 t_3^4 t_4 + 4t_1 t_3^3 t_4 + t_2 t_4^3 + 3t_3^2 t_4 + 2t_3 t_4 + t_4$$

Posinomios	Conjunto Índice	Coeficientes	Exponentes
$t_1 t_3^5 + 3t_1 t_3 t_4^2 + 3t_2 t_3^2 t_4^2 + t_4^2 + t_3^4 + t_3^3 + t_3^2 + t_3 + 1$	IG(0) = {1,2,3,4,5,6,7,8,9}	$c_1 = 1$ $c_2 = 3$ $c_3 = 3$ $c_4 = 1$ $c_5 = 1$ $c_6 = 1$ $c_7 = 1$ $c_8 = 1$ $c_9 = 1$	$\alpha_1 = (1,0,5,0)$ $\alpha_2 = (1,0,1,2)$ $\alpha_3 = (0,1,2,2)$ $\alpha_4 = (0,0,0,2)$ $\alpha_5 = (0,0,4,0)$ $\alpha_6 = (0,0,3,0)$ $\alpha_7 = (0,0,2,0)$ $\alpha_8 = (0,0,1,0)$ $\alpha_9 = (0,0,0,0)$
$t_2 t_3^4 t_4 + 4t_1 t_3^3 t_4 + t_2 t_4^3 + 3t_3^2 t_4 + 2t_3 t_4 + t_4$	IH(0) = {10,11,12,13,14,15}	$c_{10} = 1$ $c_{11} = 4$ $c_{12} = 1$ $c_{13} = 3$ $c_{14} = 2$ $c_{15} = 1$	$\alpha_{10} = (0,1,4,1)$ $\alpha_{11} = (1,0,3,1)$ $\alpha_{12} = (0,1,0,3)$ $\alpha_{13} = (0,0,2,1)$ $\alpha_{14} = (0,0,1,1)$ $\alpha_{15} = (0,0,0,1)$

De donde se observa que la función objetivo tiene 6 funciones cóncavas:

$$h_{10}(z_1, z_2, z_3, z_4) = -e^{z_2+z_3+4z_4}$$

$$h_{12}(z_1, z_2, z_3, z_4) = -4e^{z_1+3z_3+z_4}$$

$$h_{13}(z_1, z_2, z_3, z_4) = -e^{z_2+3z_4}$$

$$h_{14}(z_1, z_2, z_3, z_4) = -3e^{2z_3+z_4}$$

$$h_{15}(z_1, z_2, z_3, z_4) = -2e^{z_3+z_4}$$

$$h_{16}(z_1, z_2, z_3, z_4) = -e^{z_4}$$

El primer subproblema correspondiente al problema de optimización 1, se construirá mediante la sobrestimación de las funciones cóncavas anteriores sobre el conjunto S^1, utilizando la fórmula:

$$\ell_i^t(z) = \frac{c_i}{M_i^t - m_i^t}\left[(M_i^t e^{m_i^t} - m_i^t e^{M_i^t}) + (e^{M_i^t} - e^{m_i^t})(\alpha^i, z)\right]$$

donde:

$$S^1 = \left\{z : m_i^1 \le (\alpha^i, z) \le M_i^1 ; i \in IH\right\}$$

y

$$m_i^1 = \min_z\left\{(\alpha^i, z) : z \in R\right\}$$

$$M_i^1 = \max_z\left\{(\alpha^i, z) : z \in R\right\}$$

para $i \in IH$. Recordemos que estas cotas son fácilmente calculables si tenemos en cuenta que:

$$m_i^1 = \sum_{j=1}^{m} \min\left\{\alpha_{ij} z_j^{min}, \alpha_{ij} z_j^{max}\right\}$$

$$M_i^1 = \sum_{j=1}^{m} \max\left\{\alpha_{ij} z_j^{min}, \alpha_{ij} z_j^{max}\right\}$$

para $i \in IH$.

Si tenemos en cuenta que el conjunto R para este caso viene determinado por:

$$z_1 \in [\ln \bar{x}_{1(k-L)}, \ln \bar{x}_{1(k-L)}]$$

$$z_2 \in [\ln \bar{x}_{2(k-L)}, \ln \bar{x}_{2(k-L)}]$$

$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$

$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

y si tomamos como cotas de los estados del sistema las cotas en el estado k = 3, de forma que con el estado actual será k = 8 si la ventana temporal L = 5,

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$

$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$

i	m_i^1	M_i^1
10	1.6674	1.8632
11	1.7736	1.9751
12	-0.7279	-0.5134
13	0.1305	0.2531
14	-0.2015	-0.1073
15	-0.5336	-0.4676

Por lo tanto:

i	$\ell_i^1(z)$
10	$5.8526(z_2+z_3+4z_4)-4.4603$
11	$26.1105(z_1+3z_3+z_4)-22.7413$
12	$0.5386(z_2+3z_4)+0.8750$
13	$3.6364(2z_2+z_4)+2.9436$
14	$1.7145(z_3+z_4)+1.9805$
15	$0.6063(z_4)+0.9100$

De forma que el problema de optimización 1 quedará de la siguiente manera:

Problema de Optimización asociado a la variable x_1: (Problema Q^1)

$$\min \quad f_1^1(z_1,z_2,z_3,z_4) = g_1(z_1,z_2,z_3,z_4) - h_1^1(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$
$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$
$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$
$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4) = e^{z_1+5z_3} + 3e^{z_1+z_3+2z_4} + 3e^{z_2+2z_3+2z_4} + e^{2z_4} + e^{4z_3} + e^{3z_3} + e^{2z_3} + e^{z_3} + 1$$

$$h_1^1(z_1,z_2,z_3,z_4) = \ell_{10}^1(z_1,z_2,z_3,z_4) + \ell_{11}^1(z_1,z_2,z_3,z_4) + \ell_{12}^1(z_1,z_2,z_3,z_4) + \ell_{13}^1(z_1,z_2,z_3,z_4) + \ell_{14}^1(z_1,z_2,z_3,z_4) + \ell_{15}^1(z_1,z_2,z_3,z_4)$$

Al resolver este problema utilizando la toolbox de optimización de MATLAB obtenemos:

$$z^1 = (1.3306, 0.8895, 0.3320, -0.4676)$$

que se trata de una solución factible. Entonces:

$$v^1 = f_1^1(z^1) = 4.6749$$
$$V^1 = f_1(z^1) = 4.8394$$
$$v_b(1) = v^1 = 4.6749$$
$$V_b(1) = V^1 = 4.8394$$

Puesto que $v_b(1) < V_b(1)$ el algoritmo debe continuar. Para ello crearemos dos subproblemas a partir del término cuya diferencia entre su expresión exacta y aproximada sea mayor:

i	$\ell_i^1(z^1) - c_i e^{(\alpha^i, z^1)}$
10	0.0271
11	**0.1289**
12	0.0000
13	0.0067
14	0.0016
15	0.0000

Por lo tanto la mayor diferencia se produce para el término 11. Este será pues el término de bifurcación a escoger.

En primer lugar crearemos el problema Q^2, actualizando las cotas m_i^2 y M_i^2 a partir de los valores m_i^1 y M_i^1, excepto M_{11}^2 que se fijará a (α^{11}, z^1)=1.8590:

i	m_i^2	M_i^2
10	1.6674	1.8632
11	1.7736	**1.8590**
12	-0.7279	-0.5134
13	0.1305	0.2531
14	-0.2015	-0.1073
15	-0.5336	-0.4676

de forma que las aproximaciones de los términos se transformarán a:

i	$\ell_i^2(z)$
10	$5.8526(z_2 + z_3 + 4z_4) - 4.4603$
11	$\mathbf{24.6031}(z_1 + 3z_3 + z_4) - \mathbf{20.0680}$
12	$0.5386(z_2 + 3z_4) + 0.8750$
13	$3.6364(2z_2 + z_4) + 2.9436$
14	$1.7145(z_3 + z_4) + 1.9805$
15	$0.6063(z_4) + 0.9100$

Problema de Optimización asociado a la variable x_1: (Problema Q^2)

$$\min \quad f_1^2(z_1,z_2,z_3,z_4) = g_1(z_1,z_2,z_3,z_4) - h_1^2(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$
$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$
$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$
$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4) = e^{z_1+5z_3} + 3e^{z_1+z_3+2z_4} + 3e^{z_2+2z_3+2z_4} + e^{2z_4} + e^{4z_3} + e^{3z_3} + e^{2z_3} + e^{z_3} + 1$$

$$h_1^2(z_1,z_2,z_3,z_4) = \ell_{10}^2(z_1,z_2,z_3,z_4) + \ell_{11}^2(z_1,z_2,z_3,z_4) + \ell_{12}^2(z_1,z_2,z_3,z_4) + \ell_{13}^2(z_1,z_2,z_3,z_4) + \ell_{14}^2(z_1,z_2,z_3,z_4) + \ell_{15}^2(z_1,z_2,z_3,z_4)$$

Si resolvemos este problema Q^2 utilizando la toolbox de optimización de MATLAB se obtiene:

$$z^2 = (1.3110,\ 0.8895,\ 0.3320,\ -0.4676)$$

que se trata de una solución factible. Entonces:

$$v^2 = f_1^2(z^2) = 4.7795$$
$$V^2 = f_1(z^2) = 4.8309$$

A continuación crearemos el problema Q^3, actualizando las cotas m_i^3 y M_i^3 a partir de los valores m_i^1 y M_i^1, excepto m_{11}^3 que se fijará a (α^{11}, z^1)=1.8590:

i	m_i^3	M_i^3
10	1.6674	1.8632
11	**1.8590**	1.9751
12	-0.5130	-0.5134
13	0.1305	0.2531
14	-0.2015	-0.1073
15	-0.5336	-0.4676

de forma que las aproximaciones de los términos se transformarán a:

i	$\ell_i^3(z)$
10	$5.8526(z_2+z_3+4z_4)-4.4603$
11	$\mathbf{27.2194}(z_1+3z_3+z_4)-\mathbf{24.9317}$
12	$0.5387(z_2+3z_4)+0.8751$
13	$3.6364(2z_2+z_4)+2.9436$
14	$1.7145(z_3+z_4)+1.9805$
15	$0.6063(z_4)+0.9100$

Problema de Optimización asociado a la variable x_1: (Problema Q^3)

$$\min \quad f_1^3(z_1,z_2,z_3,z_4) = g_1(z_1,z_2,z_3,z_4) - h_1^3(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$
$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$
$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$
$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4) = e^{z_1+5z_3} + 3e^{z_1+z_3+2z_4} + 3e^{z_2+2z_3+2z_4} + e^{2z_4} + e^{4z_3} + e^{3z_3} + e^{2z_3} + e^{z_3} + 1$$

$$h_1^3(z_1,z_2,z_3,z_4) = \ell_{10}^3(z_1,z_2,z_3,z_4) + \ell_{11}^3(z_1,z_2,z_3,z_4) + \ell_{12}^3(z_1,z_2,z_3,z_4) + \ell_{13}^3(z_1,z_2,z_3,z_4) + \ell_{14}^3(z_1,z_2,z_3,z_4) + \ell_{15}^3(z_1,z_2,z_3,z_4)$$

Si resolvemos este problema Q^3 utilizando la toolbox de optimización de MATLAB se obtiene:

$$z^3 = (1.3618,\ 0.8895,\ 0.3320,\ -0.4676)$$

que se trata de una solución factible. Entonces:

$$v^3 = f_1^3(z^3) = 4.0567$$
$$V^3 = f_1(z^3) = 4.8532$$

Por lo tanto:

$$v_b(2) = 4.7795$$
$$V_b(2) = 4.8309$$

Puesto que $v_b(2) < V_b(2)$ el algoritmo debería continuar.

Para continuar deberíamos determinar el término para el cual la diferencia entre su expresión aproximada y exacta sea mayor:

i	$\ell_i^2(z^2) - c_i e^{(\alpha^i, z^1)}$
10	**0.0271**
11	0.0159
12	0.0000
13	0.0067
14	0.0016
15	0.0000

Por lo tanto la mayor diferencia se produce para el término 10. Este será pues el término de bifurcación a escoger.

En primer lugar crearemos el problema Q^4, actualizando las cotas m_i^4 y M_i^4 a partir de los valores m_i^2 y M_i^2, excepto M_{10}^4 que se fijará a (α^{11}, z^1)=1.7499:

i	m_i^4	M_i^4
10	1.6674	**1.7499**
11	1.7736	1.8590

12	-0.7279	-0.5134
13	0.1305	0.2531
14	-0.2015	-0.1073
15	-0.5336	-0.4676

de forma que las aproximaciones de los términos se transformarán a:

i	$\ell_i^4(z)$
10	$\mathbf{5.5232}(z_2+z_3+4z_4)-\mathbf{3.9110}$
11	$24.6031(z_1+3z_3+z_4)-20.0680$
12	$0.5386(z_2+3z_4)+0.8750$
13	$3.6364(2z_2+z_4)+2.9436$
14	$1.7145(z_3+z_4)+1.9805$
15	$0.6063(z_4)+0.9100$

Problema de Optimización asociado a la variable x_1: (Problema Q^4)

min $f_1^4(z_1,z_2,z_3,z_4)=g_1(z_1,z_2,z_3,z_4)-h_1^4(z_1,z_2,z_3,z_4)$

sujeto a:

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$
$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$
$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$
$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4)=e^{z_1+5z_3}+3e^{z_1+z_3+2z_4}+3e^{z_2+2z_3+2z_4}+e^{2z_4}+e^{4z_3}+e^{3z_3}+e^{2z_3}+e^{z_3}+1$$
$$h_1^4(z_1,z_2,z_3,z_4)=\ell_{10}^4(z_1,z_2,z_3,z_4)+\ell_{11}^4(z_1,z_2,z_3,z_4)+\ell_{12}^4(z_1,z_2,z_3,z_4)+$$
$$\ell_{13}^4(z_1,z_2,z_3,z_4)+\ell_{14}^4(z_1,z_2,z_3,z_4)+\ell_{15}^4(z_1,z_2,z_3,z_4)$$

Si resolvemos este problema Q^4 utilizando la toolbox de optimización de MATLAB se obtiene:

$$z^4 = (1.3110, 0.8895, 0.3320, -0.4676)$$

que se trata de una solución factible. Entonces:

$$v^4 = f_1^4(z^4) = 4.8066$$
$$V^4 = f_1(z^4) = 4.8309$$

A continuación crearemos el problema Q^5, actualizando las cotas m_i^5 y M_i^5 a partir de los valores m_i^2 y M_i^2, excepto m_{10}^5 que se fijará a (α^{11}, z^1)=1.7499:

i	m_i^5	M_i^5
10	**1.7499**	1.8632
11	1.7736	1.8590
12	-0.7279	-0.5134
13	0.1305	0.2531
14	-0.2015	-0.1073
15	-0.5336	-0.4676

de forma que las aproximaciones de los términos se transformarán a:

i	$\ell_i^5(z)$
10	$\mathbf{6.0925}(z_2+z_3+4z_4)-\mathbf{4.9073}$
11	$24.6031(z_1+3z_3+z_4)-20.0680$
12	$0.5386(z_2+3z_4)+0.8750$
13	$3.6364(2z_2+z_4)+2.9436$
14	$1.7145(z_3+z_4)+1.9805$
15	$0.6063(z_4)+0.9100$

Problema de Optimización asociado a la variable x_1: (Problema Q^5)

$$\min \quad f_1^5(z_1,z_2,z_3,z_4)=g_1(z_1,z_2,z_3,z_4)-h_1^5(z_1,z_2,z_3,z_4)$$

sujeto a:

$$z_1 \in [\ln(3.7100), \ln(3.9031)]$$
$$z_2 \in [\ln(2.3938), \ln(2.4338)]$$
$$z_3 \in [\ln(1.3938), \ln(1.4338)]$$
$$z_4 \in [\ln(0.5865), \ln(0.6265)]$$

donde:

$$g_1(z_1,z_2,z_3,z_4)=e^{z_1+5z_3}+3e^{z_1+z_3+2z_4}+3e^{z_2+2z_3+2z_4}+e^{2z_4}+e^{4z_3}+e^{3z_3}+e^{2z_3}+e^{z_3}+1$$

$$h_1^4(z_1,z_2,z_3,z_4)=\ell_{10}^5(z_1,z_2,z_3,z_4)+\ell_{11}^5(z_1,z_2,z_3,z_4)+\ell_{12}^5(z_1,z_2,z_3,z_4)+$$
$$\ell_{13}^5(z_1,z_2,z_3,z_4)+\ell_{14}^5(z_1,z_2,z_3,z_4)+\ell_{15}^5(z_1,z_2,z_3,z_4)$$

Si resolvemos este problema Q^5 utilizando la toolbox de optimización de MATLAB se obtiene:

$$z^5 = (1.3110,\ 0.8895,\ 0.3320,\ -0.4676)$$

que se trata de una solución factible. Entonces:

$$v^5 = f_1^5(z^5) = 4.8067$$
$$V^5 = f_1(z^5) = 4.8309$$

Por lo tanto:

$$v_b(3) = 4.8066$$
$$V_b(3) = 4.8309$$

Puesto que $v_b(3) < V_b(3)$ el algoritmo debería continuar. En este momento tenemos que el óptimo está acotada dentro del siguiente intervalo:

$$[4.8066, 4.8309]$$

Si deseáramos obtener un intervalo más estrecho deberíamos continuar con el algoritmo de Falk. El árbol de branch and bound construido hasta el momento es el siguiente:

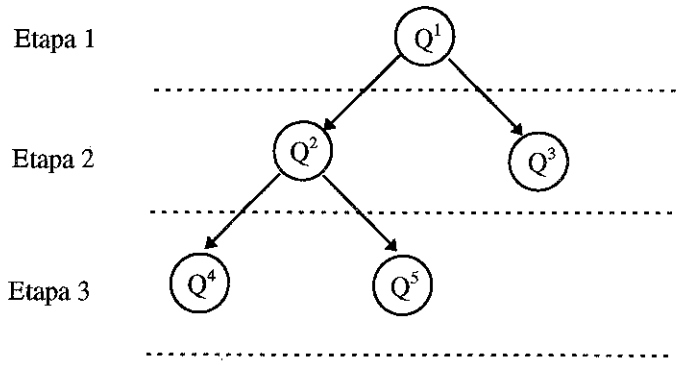


Fig. 7.7 Arbol de Branch and Bound generado por el algoritmo de Falk

El algoritmo de Falk puede llegar a determinar el óptimo de un problema signomial con la precisión deseada pero la velocidad de convergencia no es muy grande con lo cual para conseguir que las cotas v_b y V_b del óptimo se igualen, condición de finalización del algoritmo se requieren en general muchas iteraciones.

7.6 Conclusiones

En este capítulo se ha presentando la programación geométrica como una posible técnica para resolver el problema de optimización global asociado a la generación de envolventes mediante el nuevo algoritmo de generación de envolventes presentado en esta tesis.

Este algoritmo se ha escogido porque aprovecha la estructura particular del problema de optimización global a resolver: concretamente, aprovecha que la función objetivo es de tipo polinómico. En general, en optimización global no existe un algoritmo que sea capaz de resolver cualquier tipo de problema, sino que existen algoritmos que permiten resolver problemas que tengan una determinada estructura, como sucede con el algoritmo que hemos presentado en este capítulo.

Este algoritmo si bien proporciona el óptimo global con la precisión deseada, la velocidad de convergencia hacia dicho óptimo es, en general, muy lenta y además el esfuerzo de cálculo asociado es elevado puesto que a cada paso del algoritmo se generán dos subproblemas de optimización no lineal que de deben ser resueltos mediante un algoritmo de programación no lineal clásico.

A la vista del estudio realizado en este capítulo, su posible aplicación práctica en la generación de envolventes en tiempo real utilizando el nuevo algoritmo de generación de envolventes queda prácticamente descartado debido precisamente a elevado número de iteraciones y tiempo de cálculo necesario para obtener el óptimo global con suficiente precisión.

Capítulo 8

Capítulo 8:

OPTIMIZACIÓN GLOBAL APROVECHANDO LA ESTRUCTURA DEL PROBLEMA (II): PROGRAMACIÓN SEPARABLE

En este capítulo, presentaremos otro método de optimización que aprovecha la estructura particular del problema de optimización que aparece al formular el problema de generación de envolventes a partir del nuevo algoritmo basado en optimización y en la ventana deslizante. Recordemos cual es la formulación de dicho problema de optimización para cada una de las variables de estado del sistema en un determinado instante de tiempo discreto $k = n$, en función del valor de los estados en el instante de tiempo discreto $k = n\text{-}L$ donde L es la longitud de la ventana temporal, nos conduce a una problema de optimización cuya función objetivo es:

$$f(A, B, x_{n-L}) = A^{L} x_{n-L} + A^{L-1} B u_{n-L} + \cdots + B u_{n-1} \quad (8.1)$$

es una función no lineal de varias variables en concreto de: A, B y x_{n-L}. La no linealidad aparece debido a que la variable A esta elevada a potencias, y debido a los productos entre la variables A y x_{n-L} y las variables A y B. Observando la estructura de dicha función objetivo se observa que es un polinomio en el que intervienen las variables A, B y x_{n-L} multiplicadas entre si y a la vez elevadas a potencias. Dicho polinomio está formado por la suma de monomios de las mismas variables A, B y x_{n-L}, es decir, por la suma de funciones no lineales. Como veremos a continuación un problema de optimización que reúne estas características recibe el nombre de problema de optimización separable. Este tipo de problema de optimización permite ser resuelto como veremos a continuación mediante la linealización de la funciones lineales previa separación de los productos entre mediante una transformación logarítmica adecuada. Una vez separados los productos y aplicada la linealización el problema se podrá resolver mediante el método simplex, obteniéndose una solución que sólo se puede garantizar que es local, o bien, utilizando la programación entera mixta, que nos garantiza la obtención de una solución global.

Esta técnica al igual que la técnica presentada en el capítulo anterior, es una técnica de optimización global que aprovecha la estructura particular del problema, en este caso, la separabilidad de la función objetivo, mientras que la técnica anterior aprovechaba la estructura polinomial de la función objetivo. Ambas técnicas son consideradas ya como clásicas pudiéndose encontrar en cualquier libro moderno de optimización. A la vez son un buen ejemplo de que la optimización global presentada como un problema general es un problema que no puede ser resuelto por un único algoritmo, sino que en general existen multitud de algoritmos distintos que aprovechan la estructura particular de un determinado grupo de problemas, proporcionando para dicho grupo una solución aceptablemente correcta.

8.1 Programación Separable [Bazaraa93]

Un problema de optimización se denomina separable, si tanto la función objetivo como las funciones restricción se pueden expresar como suma de funciones de una única variable. Este tipo de problemas obedece a la siguiente estructura:

$$\begin{aligned} &\max\{\min\} \sum_{j=1}^{n} f_j(x_j) \quad j = 1, \cdots, n \\ &\sum_{j=1}^{n} g_{ij}(x_j)\{\leq, =, \geq\} = b_i \quad i = 1, \cdots, m \qquad (8.2) \\ &a_j \leq x_j \leq b_j \end{aligned}$$

La técnica básica de resolución de este tipo de problemas consiste en reemplazar las funciones restricción $g_{ij}(x)$ y $f_j(x_j)$ por una aproximación lineal a trozos de las mismas, de forma que el problema resultante pueda ser resuelto mediante el método simplex utilizado en programación lineal.

8.2 Aproximación Lineal a Trozos [Bazaraa93]

Consideremos una función continua arbitraria h(x) de una variable definida en el intervalo, $a \le x \le b$. Si seleccionamos r+1 puntos x_k, de forma que $x_o = a < x_1 < x_2 < ... < x_r = b$, en dicho intervalo, sin necesidad que dichos puntos sean equiespaciados. Para cada uno de los puntos escogidos calculamos $h_k = h(x_k)$ y unimos cada par de puntos (x_k,h_k) y (x_{k+1},h_{k+1}) mediante una recta, obtendremos una función lineal a trozos $\hat{h}(x)$ que aproxima a la función original h(x) en el intervalo $a \le x \le b$, tal como se ilustra en la Fig. 8.1. Dicha aproximación se puede hacer tan buena como se quiera escogiendo los puntos x_k de forma adecuada y subdividiendo el intervalo $a \le x \le b$ tan finamente como se desee.

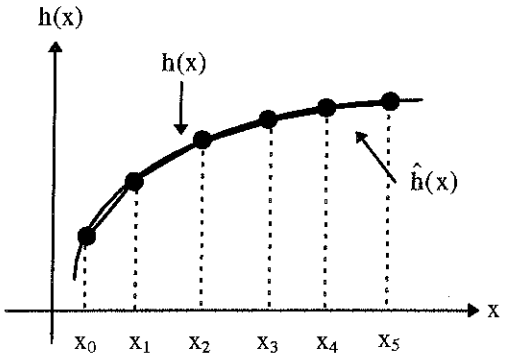


Fig. 8.1 Aproximación lineal a trozos de una función

Vamos aplicar esta idea sobre el problema de optimización separable presentado en el apartado anterior. Supongamos que en dicho problema las funciones objetivo $f_j(x)$ y restricción $g_{ij}(x)$ son continuas, si subdividimos el intervalo sobre el está definida cada variable x_j utilizando una rejilla de puntos x_{kj}, utilizando el método que acabamos de presentar, podremos determinar las correspondientes aproximaciones lineales para $\hat{f}_j(x)$ y $\hat{g}_{ij}(x)$. Utilizando dichas aproximaciones el problema de optimización original se transforma en:

$$\max\{\min\} \sum_{j=1}^{n} \hat{f}_j(x_j) \quad j = 1, \cdots, n$$

$$\sum_{j=1}^{n} \hat{g}_{ij}(x_j) \{\le, =, \ge\} b_i \quad i = 1, \cdots, m \qquad (8.3)$$

$$a_j \le x_j \le b_j$$

al que denominaremos **problema aproximado**. Una formulado dicho problema vamos a concentramos en su solución. Para ello en primer lugar, determinaremos la expresión analítica de las funciones aproximadas $\hat{f}_j(x)$ y $\hat{g}_{ij}(x)$.

- **Aproximación utilizando puntos absolutos: forma λ**

Observando la Fig. 8.1 se puede ver que en el intervalo $x_k \le x \le x_{k+1}$, se está aproximando h(x) por $\hat{h}(x)$, donde

$$\hat{h}(x) = h_k + \frac{h_{k+1} - h_k}{x_{k+1} - x_k}(x - x_k) \qquad (8.4)$$

De forma que, cualquier x en el intervalo $x_k \le x \le x_{k+1}$ se puede escribir como

$$x = \lambda x_{k+1} + (1-\lambda)x_k \qquad (8.5)$$

para cualquier λ, tal que $0 \le \lambda \le 1$. De donde

$$(x - x_k) = \lambda(x_{k+1} - x_k) \qquad (8.6)$$

y entonces

$$\hat{h}(x) = \lambda h_{k+1} + (1-\lambda)h_k \qquad (8.7)$$

Si hacemos que $\lambda = \lambda_{k+1}$ y $(1-\lambda) = \lambda_k$, entonces en el intervalo $x_k \le x \le x_{k+1}$, existen unos únicos valores de λ_k y λ_{k+1} tales que

$$\begin{aligned} x &= \lambda_k x_k + \lambda_{k+1} x_{k+1} \\ \hat{h}(x) &= \lambda_k h_k + \lambda_{k+1} h_{k+1} \\ \lambda_k &+ \lambda_{k+1} = 1 \\ \lambda_k &, \lambda_{k+1} \ge 0 \end{aligned} \qquad (8.8)$$

Y en general para $a \le x \le b$, se puede escribir

$$\begin{aligned} x &= \sum_{k=0}^{r} \lambda_k x_k \\ \hat{h}(x) &= \sum_{k=0}^{r} \lambda_k h_k \\ \sum_{k=0}^{r} &\lambda_k = 1 \\ \lambda_k &\ge 0 \quad k = 0, \cdots, r \end{aligned} \qquad (8.9)$$

donde además se requiere que no más de dos λ_k sean positivos (o sea, diferentes de cero), y si dos son positivos (o sea, diferentes de cero), por ejemplo, λ_k y λ_s con s>k, se debe cumplir que s = k + 1, o sea que sean adyacentes. Con esta restricción, los valores λ_k quedan determinados de forma única, y para un valor de x dado por

$$x = \sum_{k=0}^{r} \lambda_k x_k \qquad (8.10)$$

el valor aproximado de la función $\hat{h}(x)$ vendrá dado por la expresión analítica:

$$\hat{h}(x) = \sum_{k=0}^{r} \lambda_k h_k \qquad (8.11)$$

No permitiendo que más de dos λ_k sean positivos (o sea, diferentes de cero), y que además añadiendo la condición de adyacencia, se asegura que los puntos $(x, \hat{h}(x))$ estarán sobre la aproximación lineal. Si no existieran dichas restricciones sobre λ_k no se podría garantizar que todos los puntos estuvieran sobre la aproximación lineal.

Hasta ahora hemos visto como cualquier función continua se puede aproximar por una aproximación lineal a trozos, y a la vez hemos visto como obtener su representación matemática. Volviendo al problema de optimización original que se nos había planteado, supongamos que para cada variable x_j podemos determinar mediante consideraciones sobre la física del problema el intervalo de valores que puede tomar, o sea, $a_j \le x_j \le b_j$. Una vez determinado dicho intervalo lo subdividimos en r_j subintervalos mediante $r_j + 1$ puntos x_{kj}. Entonces, para todas las funciones $\hat{f}_j(x_j)$ y $\hat{g}_{ij}(x_j)$ podemos escribir

$$\hat{f}_j(x_j) = \sum_{k=0}^{r_j} \lambda_{kj} f_{kj} \quad \text{donde} \quad f_{kj} = f_j(x_k)$$

$$\hat{g}_{ij}(x_j) = \sum_{k=0}^{r_j} \lambda_{kj} g_{kij} \quad \text{donde} \quad g_{kij} = g_{ij}(x_k) \qquad (8.12)$$

$$\text{para } i = 1, \cdots, m$$

donde

$$x_j = \sum_{k=0}^{r_j} \lambda_k x_{kj}$$

$$\sum_{k=0}^{r_j} \lambda_{kj} = 1 \qquad (8.13)$$

$$\lambda_{kj} \geq 0 \quad \text{para todo k, j}$$

y para cualquier j, no más de dos λ_{kj} pueden ser estrictamente positivos y además los dos que sean diferentes de cero deben ser adyacentes. Debe notarse que a el escribir las expresiones analíticas de las funciones $\hat{f}_j(x_j)$ y $\hat{g}_{ij}(x_j)$ se han utilizado las mismas subdivisiones para los intervalos $a_j \leq x_j \leq b_j$. Los puntos de subdivisión de los intervalos x_{kj} deberían de ser escogidos de forma que las funciones originales $f_j(x_j)$ y $g_{ij}(x_j)$ se pudieran representar con suficiente precisión.

Utilizando las expresiones que acabamos de determinar para aproximar las funciones $f_j(x_j)$ y $g_{ij}(x_j)$ el problema de optimización original se puede expresar en función de las variables λ_{kj} de la siguiente manera

$$\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} f_{kj} \lambda_{kj}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} g_{kij} \lambda_{kj} \{\leq, =, \geq\} b_i \quad \text{para } i = 1, \cdots, m \qquad (8.14)$$

$$\sum_{k=0}^{r_j} \lambda_{kj} = 1 \quad \text{para } j = 1, \cdots, n$$

$$\lambda_{kj} \geq 0 \quad \text{para todo k, j}$$

El problema sería lineal si no fuera por el hecho de que para cada j, se requiere que no más de dos λ_{kj} sean positivos y a la vez adyacentes. Este problema se podría resolver utilizando el método simplex tradicional añadiendo como restricciones adicionales que nunca pueden existir más de dos valores λ_{kj} diferentes de cero y además deben ser adyacentes, tal como se mostrará más adelante. Una vez resuelto el problema los valores de x_j se pueden determinar utilizando $x_j = \sum_{k=0}^{r_j} \lambda_k x_{kj}$.

- **Aproximación lineal utilizando la diferencia entre puntos absolutos: forma δ**

El problema de optimización original se puede aproximar de una forma diferente utilizando un nuevo conjunto de variables. Partiendo de:

$$\hat{f}_j(x_j) = \sum_{k=0}^{r_j} \lambda_{kj} f_{kj} \quad \text{donde} \quad f_{kj} = f_j(x_k)$$

$$\hat{g}_{ij}(x_j) = \sum_{k=0}^{r_j} \lambda_{kj} g_{kij} \quad \text{donde} \quad g_{kij} = g_{ij}(x_k) \qquad (8.15)$$

$$\text{para } i = 1, \cdots, m$$

donde

$$x_j = \sum_{k=0}^{r_j} \lambda_k x_{kj}$$

$$\sum_{k=0}^{r_j} \lambda_{kj} = 1 \qquad (8.16)$$

$$\lambda_{kj} \geq 0 \quad \text{para todo } k, j$$

podemos definir:

$$\Delta f_{kj} = f_{kj} - f_{(k-1)j}$$

$$\Delta g_{kij} = g_{kij} - g_{(k-1)ij} \qquad (8.17)$$

$$\Delta x_{kj} = x_{kj} - x_{(k-1)j}$$

$$\text{para } k = 1, \cdots, r_j$$

Entonces, si x_j pertenece al intervalo $[x_{(k-1)j}, x_{kj}]$ entonces podemos escribir:

$$x_j = x_{(k-1)j} + (\Delta x_{kj})\delta_{kj} \qquad (8.18)$$

donde

$$\delta_{kj} = \frac{x_j - x_{(k-1)j}}{\Delta x_{kj}} \qquad (8.19)$$

queda unívocamente determinado por x_j. Además, para x_j en el intervalo considerado, se cumple

$$0 \leq \delta_{kj} \leq 1 \qquad (8.20)$$

A partir de

$$\hat{h}(x) = h_k + \frac{h_{k+1} - h_k}{x_{k+1} - x_k}(x - x_k) \qquad (8.21)$$

podemos ver que si x_j pertenece al intervalo $[x_{(k-1)j}, x_{kj}]$ entonces $\hat{f}_j(x_j)$ y $\hat{g}_{ij}(x_j)$ las aproximaciones lineales a trozos de $f_j(x_j)$ y $g_{ij}(x_j)$, se pueden escribir respectivamente

$$f_j(x_j) = f_{(k-1)j} + (\Delta f_{kj})\delta_{kj} \qquad (8.22)$$

$$\hat{g}_{ij}(x_j) = g_{(k-1)ij} + (\Delta g_{kij})\delta_{kj} \qquad (8.23)$$

Supongamos ahora que imponemos como restricciones adicionales que si $\delta_{kj} > 0$ entonces

$$\delta_{uj} = 1 \quad \text{para} \quad u = 1, \cdots, k-1 \qquad (8.24)$$

Para el caso que estamos considerando, podemos escribir lo siguiente

$$x_{(k-1)j} = \sum_{u=1}^{k-1} (\Delta x_{uj})\delta_{uj}$$

$$g_{(k-1)ij} = \sum_{u=1}^{k-1} (\Delta g_{uij})\delta_{uj} + g_{0ij} \qquad (8.25)$$

$$f_{(k-1)j} = \sum_{u=1}^{k-1} (\Delta f_{uj})\delta_{uj} + f_{0j}$$

Además, si $0 \leq \delta_{kj} \leq 1$, la restricción anterior implica que $\delta_{uj} = 0$, para $u > k$. Por lo tanto, si se requiere que cuando $\delta_{kj} > 0$, entonces $\delta_{uj} = 1$, para u = 1, ... , k-1 , entonces podemos escribir que

$$x_j = \sum_{k=1}^{r_j} (\Delta x_{kj})\delta_{kj}$$

$$\hat{g}_{ij}(x_j) = \sum_{k=1}^{r_j} (\Delta g_{kij})\delta_{kj} + g_{0ij} \qquad (8.26)$$

$$\hat{f}_j(x_j) = \sum_{k=1}^{r_j} (\Delta f_{kj})\delta_{kj} + f_{0j}$$

y δ_{kj} puede ser determinada de forma única cuando se cumple $0 \le \delta_{kj} \le 1$. El problema de optimización aproximado se puede expresar ahora en términos de las variables δ_{kj} en lugar de las variables λ_{kj}. En términos de las variables δ_{kj} dicho problema puede expresarse de la siguiente manera

$$\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta f_{kj})\delta_{kj} - \sum_{j=1}^{n} f_{0j}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta g_{kij})\delta_{kj} \{\le, =, \ge\} b_i - \sum_{j=1}^{n} g_{0ij} \quad \text{para } i = 1, \cdots, m \qquad (8.27)$$

$$0 \le \delta_{kj} \le 1 \quad \text{para todo k, j}$$

donde además se requiere que si $\delta_{kj} > 0$ entonces $\delta_{uj} = 1$ para u = 1, ... , k-1. El problema de optimización sería lineal si no fuera por la restricciones existentes sobre los valores de δ_{kj}. Este problema se podría resolver aplicando el método simplex tradicional, pero teniendo en cuenta que no se debería permitir que δ_{kj} fuera positivo a menos que $\delta_{uj} = 1$ para u = 1, ... , k-1.

- **Comparación entre las formas λ y δ**

A continuación vamos a comparar ambas formas. Dicha comparación se va a realizar para el caso en que cada una de las variables aparezca en el problema de optimización a través de una función no lineal. La forma λ implica $n + \sum_{j=1}^{n} r_j$ variables y m+n restricciones. La solución final implicará a no más de n+m valores positivos de λ_{kj} (distintos de cero) y para cada valor de j no más de dos valores de λ_{kj} serán distintos de cero. La forma δ implica a $\sum_{j=1}^{n} r_j$ variables y m restricciones. Además, para esta forma no se deben incluir de forma explícita $\sum_{j=1}^{n} r_j$ cotas superiores como restricciones adicionales. La solución final implicará a $m + \sum_{j=1}^{n} r_j$ valores de δ_{kj} diferentes de cero. La ventaja de la forma δ es que tiene la ventaja de que trabaja con un problema menor en cuanto al número de variables y restricciones. Ambos métodos están disponibles en muchos solvers comerciales. De la experiencia que se tiene sobre ambas formas, la forma δ precisa menos tiempo de cálculo debido al menor número de variables y restricciones, aunque el número de iteraciones puede ser ligeramente superior que para la forma λ. Así mismo, la experiencia ha demostrado que ambas formas requieren mayor tiempo de cálculo que el requerido por un problema estrictamente lineal. Por ejemplo, un problema en forma λ con 120 restricciones requiere alrededor de 1200 iteraciones antes de finalizar, mientras que un problema estrictamente lineal se resolvería sólo con unas 400 iteraciones.

8.3 Transformación de Variables para obtener Separabilidad [Bazaraa93]

Aunque un problema puede que originalmente no tenga la forma de un problema separable:

$$\max\{\min\} \sum_{j=1}^{n} f_j(x_j) \quad j=1,\cdots,n$$

$$\sum_{j=1}^{n} g_{ij}(x_j)\{\leq,=,\geq\} = b_i \quad i=1,\cdots,m \qquad (8.28)$$

$$a_j \leq x_j \leq b_j$$

definiendo una transformación de variables adecuada se puede conseguir que todas las funciones sean separables. Existen diversas técnicas.

- **Primera técnica**

Supongamos que el producto $x_i x_j$ aparece ya sea en la función objetivo ya sea en las funciones restricción. Para eliminar dicho producto y obtener una función separable, supongamos que definidos dos variables nuevas:

$$y_i = \frac{1}{2}(x_i + x_j)$$
$$y_j = \frac{1}{2}(x_i - x_j) \qquad (8.29)$$

Entonces, el producto se transformará en

$$x_i x_j = y_i^2 - y_j^2 \qquad (8.30)$$

obteniendo una función separable con dos nuevas variables y_i e y_j.

- **Segunda técnica**

El producto entre variables $x_i x_j$ puede ser eliminado utilizando otra técnica que puede separar no sólo productos de dos variables sino de hasta más de dos y a la vez cada una de dichas variables puede estar elevada a potencias, con la única restricción de que dichas variables sólo tomen valores positivos. Dicha técnica consiste en el siguiente cambio de variable:

$$y = x_i x_j \qquad (8.31)$$

para a continuación aplicar sobre dicha expresión logaritmos, de forma que:

$$\ln y = \ln x_i + \ln x_j \qquad (8.32)$$

con lo cual se consigue la separación de las variables. La restricción de que dicha técnica sólo se puede aplicar para variables que tomen valores positivos se debe a que utilizan los logaritmos de dichas variables para separarlas. Esta técnica es muy parecida a la utilizada en el capítulo anterior, dedicado a la programación signomial, donde la separabilidad se conseguía también mediante una transformación, que en aquel caso era de tipo exponencial. Ambas transformaciones la que presenta ahora y la que se presentó en el capítulo anterior aprovechan el propiedad de que tanto las funciones exponenciales como las funciones logarítmicas permiten transformar productos en sumas, propiedad por la cual se las utilizó también en la época en la que no existían calculadoras para realizar operaciones en las que intervenían productos.

Esta técnica se puede modificar para que admita que las variables x_i y x_j puedan tomar valores negativos. Supongamos que introducimos dos nuevas variables:

$$w_i = x_i + \varepsilon_i$$
$$w_j = x_j + \varepsilon_j \qquad (8.33)$$

donde ε_i y ε_j son números positivos a escoger, de forma que:

$$\begin{aligned} w_i &\geq \varepsilon_i > 0 \\ w_j &\geq \varepsilon_j > 0 \end{aligned} \qquad (8.34)$$

Por lo tanto:

$$x_i x_j = (w_i - \varepsilon_i)(w_j - \varepsilon_j) = w_i w_j - \varepsilon_i w_j - \varepsilon_j w_i + \varepsilon_i \varepsilon_j \qquad (8.35)$$

Ahora podemos aplicar el método de separación, visto anteriormente, sobre las variables w_i y w_j de la siguiente manera:

$$y = w_i w_j \qquad (8.36)$$

para a continuación tomar logaritmos, pero ahora ya sobre variables que sólo pueden tomar valores positivos:

$$\ln y = \ln w_i + \ln w_j \qquad (8.37)$$

Como conclusión vemos que para separar el producto de dos variables que pueden tomar valores negativos se deben introducir tres variables nuevas (w_i, w_j e y) junto con sus correspondientes tres restricciones.

8.4 Programación Lineal: el Método Simplex [Bazaraa93]

El método simplex es un procedimiento sistemático para resolver problemas de programación lineal a base de moverse de un punto extremo a otro punto extremo con un valor de la función objetivo mejor. Este proceso se repite hasta que se alcanza un punto óptimo, o bien, hasta que se determina que el valor de la función objetivo no está acotado.
Consideremos un problema de programación lineal en formato estándar:

$$\begin{aligned} &\min c^t x \\ &\text{s.t. } Ax = b \\ &\quad x \geq 0 \end{aligned} \qquad (8.38)$$

Si el problema no estuviera en forma estándar, siempre se podría transformar utilizando variables de holgura ("slack variables"). Así por ejemplo, si el problema tuviera la siguiente restricción

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (8.39)$$

se puede transformar a la forma estándar añadiéndole la **variable de holgura** no negativa s_i, de forma que se transforma en

$$\sum_{j=1}^{n} a_{ij} x_j + s_i = b_i \qquad (8.40)$$

Supongamos que tenemos un punto extremo $\bar{x}$. Este punto se caracteriza por una descomposición de A en [B,N], donde $B = [a_{B_1}, \cdots, a_{B_m}]$ es una matriz $m \times m$ de rango completo denomina **base**, y N es una matriz $m \times (n-m)$, de forma que $\bar{x}$ se puede escribir de la siguiente forma:

$$\bar{x}^t = (\bar{x}_B^t, \bar{x}_N^t) = (\bar{b}, 0^t) \qquad (8.41)$$

donde:

$$\bar{b} = B^{-1} b \geq 0 \qquad (8.42)$$

Las variables que corresponden a la base B se denominan **variables básicas** y se indican mediante $x_{B_1}, \cdots, x_{B_m}$ mientras que las variables que corresponden a N se denominan **variables no básicas**.

- **Búsqueda de una mejor solución**

Consideremos ahora un punto x que satisfaga: Ax = b y x ≥ 0. Si descomponemos x^t en (x_B^t, x_N^t) y observamos que x_B, $x_N \geq 0$. Así mismo, Ax = b se puede escribir como:

$$Bx_B + Nx_N = b \qquad (8.43)$$

Por lo tanto:

$$x_B = B^{-1}b - B^{-1}Nx_N \qquad (8.44)$$

Entonces:

$$\begin{aligned} c^t x &= c_B^t x_B + c_N^t x_N \\ &= c_B^t B^{-1} b + (c_N^t - c_B^t B^{-1} N) x_N \\ &= c^t \bar{x} + (c_N^t - c_B^t B^{-1} N) x_N \end{aligned} \qquad (8.45)$$

De donde, si $c_N^t - c_B^t B^{-1} N \geq 0$, puesto que $x_N \geq 0$, se cumple que: $c^t x \geq c^t \bar{x}$ y por lo tanto $\bar{x}$ es punto extremo.

En particular, supongamos que la j-ésima componente $c_j - c_B^t B^{-1} a_j$ es negativa. Consideremos:

$$x = \bar{x} + \lambda d_j \qquad (8.46)$$

donde:

$$d_j = \begin{pmatrix} -B^{-1}a_j \\ e_j \end{pmatrix} \qquad (8.47)$$

donde e_j es un vector unitario de n-m componentes y con un 1 en la posición j-ésima. Entonces, a partir de las expresiones anteriores:

$$c^t x = c^t \bar{x} + \lambda (c_j - c_B^t B^{-1} a_j) \qquad (8.48)$$

consiguiendo $c^t x < c^t \bar{x}$ para $\lambda > 0$ puesto que $c_j - c_B^t B^{-1} a_j < 0$, debiendo considerar los dos casos siguientes en función del valor de $y_j = B^{-1} a_j$

Caso 1: ($y_j \leq 0$)

Obsérvese que $Ad_j = 0$, y puesto que $A\bar{x} = b$, entonces Ax = b para $x = \bar{x} + \lambda d_j$ y para todos los valores de λ. Por lo tanto, x es factible si y sólo si x ≥ 0. Evidentemente, esta condición se cumple para todo λ ≥ 0 si $y_j \leq 0$. Por lo tanto, utilizando $c^t \bar{x} + \lambda (c_j - c_B^t B^{-1} a_j)$ se puede comprobar que para este caso el valor de la función objetivo no está acotado. Es decir, se ha determinado una dirección d_j para la cual $c^t d_j = c_j - c_B^t B^{-1} a_j < 0$.

Caso 2: ($y_j > 0$)

Sea $B^{-1}b = \bar{b}$ y sea λ definido mediante

$$\lambda = \min_{1 \leq i \leq m} \left\{ \frac{\bar{b}_i}{y_{ij}} : y_{ij} > 0 \right\} = \frac{\bar{b}_r}{y_{rj}} \geq 0 \qquad (8.49)$$

donde y_{ij} es la i-ésima componente de y_j. En este caso, las componentes de $x = \bar{x} + \lambda d_j$ vienen dadas por

$$x_{B_i} = \bar{b}_i - \frac{\bar{b}_r}{y_{rj}} y_{ij} \quad \text{para } i = 1, \cdots, m$$
$$x_j = \frac{\bar{b}_r}{y_{rj}} \tag{8.50}$$

y todas las otras x_i son igual a cero. Las componentes de x distintas de cero sólo pueden ser $x_{B_1}, \cdots, x_{B_{r-1}}, x_{B_{r+1}}, \cdots, x_{B_m}$ y x_j. Por lo tanto, como máximo puede haber m componentes de x diferentes de cero, siendo fácil comprobar si sus correspondientes columnas en A son linealmente independientes. Por lo tanto, el punto x es también un punto extremo. En este caso, decimos que la variable básica x_B abandona la base y en su lugar la variable no básica x_j entra en la base.

Hasta ahora hemos visto pues, que dado un punto extremo, se puede comprobar si es óptimo y por lo tanto no continuar buscando, o bien, determinar una dirección que nos puede conducir a una solución no acotada, o a una solución mejor.

- **El algoritmo simplex**

A. Inicialización

En primer lugar se debe determinar un punto extremo x con su base B. Si no se dispone de dicho punto, entonces existen dos procedimientos posibles para obtenerla:

Opción 1: "Método de las Dos Fases"

En este método, las restricciones del problema se alteran utilizando variables artificiales de forma que sea fácil obtener un punto extremo del nuevo sistema. Concretamente, las restricciones se modifican de la siguiente manera:

$$\begin{aligned} Ax + x_a &= b \\ x, x_a &\geq 0 \end{aligned} \tag{8.51}$$

donde x_a es un vector artificial. Entonces, $x = 0$ y $x_a = b$ constituye un punto extremo del anterior sistema. Puesto que solo se obtendrá una solución factible del problema original cuando $x_a = 0$, se puede utilizar el método simplex para minimizar la suma de las variables artificiales partiendo de su valor extremo original. Esto nos lleva a resolver el siguiente problema en la Fase 1:

$$\begin{aligned} &\min e^t x_a \\ &\text{s.t. } Ax + x_a = b \\ &\quad x, x_a \geq 0 \end{aligned} \tag{8.52}$$

donde e es un vector de unos. Al final de la Fase 1, tendremos que $x_a \neq 0$, o bien, $x_a = 0$. En el primer caso, se concluirá que el sistema original es inconsistente, o sea, que la región factible está vacía. En el segundo caso, las variables artificiales habrán desaparecido de la base, y por lo tanto, habremos obtenido un punto extremo del sistema original. Utilizando este punto extremo, la Fase 2 del algoritmo simplex minimizará la función objetivo original.

Opción 2: "Método de la M"

De la misma forma que en el método de las dos fases, en este método se modifican las restricciones introduciendo variables artificiales de forma que el punto extremo del nuevo problema se obtenga de forma inmediata. A continuación se asigna un coeficiente de coste M a cada una de las variables artificiales de forma que hará que su valor caiga a cero rápidamente. Esto nos lleva a resolver el siguiente problema:

$$\begin{aligned} &\min c^t x + Me^t x_a \\ &\text{s.t. } Ax + x_a = b \\ &x, x_a \geq 0 \end{aligned} \qquad (8.53)$$

A continuación se puede aplicar el método simplex, sobre dicho problema, sin especificar un valor numérico para el coeficiente M simplemente manejando los coeficientes de la función objetivo correspondientes a M para las variables no básicas como un vector separado. Si al terminar la aplicación del método simplex, $x_a = 0$, entonces se ha alcanzado la solución del problema original. En caso contrario, si $x_a \neq 0$, se puede concluir que el problema original no tiene solución factible.

B. Ciclo principal

Paso 1

Sea x un punto extremo con base B. Entonces se debe calcular: $c_B^t B^{-1} N - c_N^t$. Si el resultado es no positivo, entonces el algoritmo debe finalizar: x es un punto extremo óptimo. En caso contrario, debemos escoger la componente positiva mayor de $c_B^t B^{-1} a_j - c_j$. Si $y_j = B^{-1} a_j \leq 0$, entonces el algoritmo debe finalizar, el valor de la función objetivo no está acotado a lo largo de la dirección:

$$\left\{ x + \lambda \begin{bmatrix} -y_j \\ e_j \end{bmatrix} : \lambda \geq 0 \right\} \qquad (8.54)$$

donde e_j es un vector de ceros excepto en la posición j-ésima donde tiene un 1. Por otro laso si: $y_j > 0$, iremos al paso 2.

Paso 2

Determinaremos el índice r a partir de:

$$\lambda = \min_{1 \leq i \leq m} \left\{ \frac{\bar{b}_i}{y_{ij}} : y_{ij} > 0 \right\} = \frac{\bar{b}_r}{y_{rj}} \geq 0 \qquad (8.55)$$

donde y_{ij} es la i-ésima componente de y_j. En este caso, las componentes de $x = \bar{x} + \lambda d_j$ vienen dadas por

$$\begin{aligned} x_{B_i} &= \bar{b}_i - \frac{\bar{b}_r}{y_{rj}} y_{ij} \quad \text{para } i = 1, \cdots, m \\ x_j &= \frac{\bar{b}_r}{y_{rj}} \end{aligned} \qquad (8.56)$$

y todas las otras x_i son igual a cero. A continuación formaremos la nueva base borrando la columna a_B e introduciendo a_j en su lugar. Finalmente volveremos al paso 1.

- **Formato Tableau del método simplex**

Supongamos que tenemos la base inicial B correspondiente a un punto extremo inicial. La función objetivo y las restricciones se pueden escribir de la siguiente manera:

Fila Objetivo: $f - c_B^t x_B - c_N^t = 0$
Filas Restricción: $Bx_B + Nx_N = b$

Estas ecuaciones se pueden mostrar en el siguiente "tableau simplex", donde las entradas en la columna RHS son las constantes que aparecen en la parte derecha de las ecuaciones:

f	x_B^t	x_N^t	RHS
1	$-c_B^t$	$-c_N^t$	0
0	B	N	b

Las filas restricción se actualizan multiplicando por B^{-1}, mientras que las filas objetivo se actualizan añadiéndoles $c_B{}^t$ veces las nuevas filas restricción, obteniendo el siguiente tableau actualizado $\overline{b}$

	f	x_B^t	x_N^t	RHS
f	1	0^t	$c_B^t B^{-1}N - c_N^t$	$c_B^t \overline{b}$
x_B	0	I	$B^{-1}N$	$\overline{b}$

Debe observarse que las variables básicas se han colocado en la parte izquierda, y que $\overline{b} = B^{-1}b$. Por otro lado los valores de las variables básicas y de la función objetivo f se han colocado en la parte derecha del tableau. Así mismo, el vector $c_B^t B^{-1}N - c_N^t$ y la matriz $B^{-1}N$ se han almacenado debajo de las variables no básicas.

El tableau que acabamos de presentar muestra toda la información necesaria para realizar el paso 1 del algoritmo simplex:

1. Si $c_B^t B^{-1}N - c_N^t \leq 0$, entonces el algoritmo finalizará, puesto se ha llegado al punto óptimo.

2. En caso contrario, examinaremos la fila objetivo, escogiendo una variable no básica que cumpla $c_B^t B^{-1}a_j - c_j \leq 0$. Si $y_j = B^{-1}a_j \leq 0$ entonces el algoritmo finalizará, puesto que el problema tiene una función objetivo no acotada.

3. Por otro lado, si $y_j = B^{-1}a_j > 0$, puesto que $\overline{b}$ e y_j están almacenadas debajo de RHS y x_j respectivamente, entonces el valor de λ dado por

$$\lambda = \min_{1 \leq i \leq m} \left\{ \frac{\overline{b}_i}{y_{ij}} : y_{ij} > 0 \right\} = \frac{\overline{b}_r}{y_{rj}} \geq 0 \qquad (8.57)$$

se puede calcular fácilmente utilizando los datos del tableau. La variable básica x_{Br}, correspondiente al mínimo valor dado por la expresión anterior deberá abandonar la base dejando paso a la variable x_j.

4. Finalmente, el tableau se deberá actualizar para reflejar la nueva base. Esta actualización puede realizarse pivotando entre la fila x_{Br} y la columna x_j de la siguiente manera:

   (a) Se debe dividir la r-ésima fila correspondiente a x_{Br} por y_{rj}.
   (b) Multiplicar la nueva r-ésima fila por y_{ij} a la vez que se resta dicha cantidad de la i-ésima fila para i=1,...,m con $i \neq r$
   (c) Multiplicar la nueva r-ésima fila por $c_B^t B^{-1}a_j - c_j$ a la vez que se resta dicha cantidad de la fila objetivo.

8.5 Solución del Problema Separable Linealizado mediante Simplex [Bazaraa93]

El problema que se desea resolver es el problema de optimización separable linealizado:

$$\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} f_{kj}\lambda_{kj}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} g_{kij}\lambda_{kj} \{\leq, =, \geq\} b_i \quad \text{para } i = 1, \cdots, m \tag{8.58}$$

$$\sum_{k=0}^{r_j} \lambda_{kj} = 1 \quad \text{para } j = 1, \cdots, n$$

$$\lambda_{kj} \geq 0 \quad \text{para todo } k, j$$

Tal como hemos comentado anteriormente este problema seria un problema lineal si fuera porque para cada valor de j, se requiere que no más de dos valores λ_{kj} sean distintos de cero y además adyacentes. Para resolver el problema se propone utilizar el método simplex, utilizado para problemas lineales, de la forma tradicional pero restringiendo la entrada a la base de tal manera que se cumpla que mas de dos valores λ_{kj} sean diferentes de cero para una valor de j dado, y además se cumpla la condición de adyacencia.

Después de añadir las variables de holgura al problema separable linealizado, se puede escribir de la siguiente manera:

$$\max\{\min\} \hat{z} = f\lambda$$

$$\text{s.t.} \quad G\lambda = b \tag{8.59}$$

$$\lambda \geq 0$$

donde la matriz G contiene m + n filas y $\sum_j r_j$ + s + n columnas (siendo s el número de variables de holgura que se han añadido.
Se puede demostrar que cuando se utiliza el método simplex con la entrada a la base restringida se alcanza un punto en el cual:

$$z_{kj} - f_{kj} \geq 0 \tag{8.60}$$

para cualquier valor de λ_{kj} que pueda entrar la base. Entonces, podemos afirmar que se ha alcanzado un máximo relativo del problema separable aproximado con respecto a las variables x_j. Dicho de otra forma, si x_0 es el valor de x que corresponde a la solución final básica λ_0, habiéndose obtenido x_0 a partir de λ_0 mediante $x_j = \sum_{k=0}^{r_j} \lambda_k x_{kj}$

- **Crítica al Método Simplex: problemas con los óptimos locales**

El método de aproximar linealmente un problema de optimización no lineal separable nos ha permitido abordarlo con técnicas propias de problemas de optimización lineal, es decir, con el algoritmo simplex. En general, utilizando esta técnica sólo seremos capaces de determinar un óptimo local (máximo o mínimo) para el problema aproximado y por lo tanto un óptimo local aproximado para el problema original. Sólo cuando las funciones $f_j(x_j)$ y $g_{ij}(x_j)$ posean las propiedades adecuadas de convexidad o concavidad podremos asegurar que se ha alcanzado un óptimo global para el problema aproximado, y por lo tanto, una aproximación al óptimo global para el problema original.

8.6 Programación Entera Mixta aplicada a problemas separables linealizados
[Fuente95]

Hasta el momento en este capítulo se ha presentado la técnica de linealización a trozos como un posible método de solución de problemas de optimización no lineales separables. El algoritmo propuesto para resolver el problema linealizado resultante, el algoritmo simplex, no puede garantizar que la solución del problema aproximado sea el óptimo global. Sólo en determinados casos particulares cuando las funciones objetivo y restricción reúnen determinadas propiedades de convexidad/concavidad. En este apartado se propone la utilización de la programación entera mixta como una técnica alternativa al algoritmo simplex para poder resolver el problema separable linealizado garantizando que el óptimo calculado para dicho problema es el óptimo global.

- **Introducción a la programación entera mixta**

En primer lugar, vamos a presentar los problemas de programación lineales enteros. Un problema de programación lineal entero es un problema de programación no lineal que sería lineal si no fuera por el hecho de que algunas o todas las variables sólo pueden tomar valores enteros. El problema de denomina **problema entero** si todas las variables sólo pueden tomar valores enteros, mientras que se denomina **problema entero mixto** si sólo algunas de las variables deben tomar valores enteros. Un problema entero en general puede escribirse de la siguiente manera:

$$\begin{aligned} &\min\{\max\}\ cx \\ &\text{s.t. } Ax = b \\ &\quad x \geq 0 \end{aligned} \qquad (8.61)$$

donde x_j para $j \in J$ es una variables entera. Si J es un conjunto vacío, entonces el problema de optimización es un problema de programación lineal.

- **Solución de un problema separable linealizado**

Al principio de este capítulo se ha visto como se podía transformar un problema de optimización no lineal en un problema lineal a trozos. A continuación se ha presentado el algoritmo simplex con restricción de entrada en la base como un posible método de solución del problema aproximado obtenido. El problema de esta técnica es que sólo puede garantizar un óptimo local del problema aproximado, excepto para situaciones particulares en las cuales las funciones objetivo y restricción posean ciertas propiedades de convexidad/concavidad. En este apartado vamos a ver como se puede formular el problema de optimización aproximado como un problema de programación entera mixta. Resolviendo dicho problema se obtendrá para cualquier tipo de funciones objetivo y restricción el óptimo global.

A. Forma δ

Empezaremos en primer lugar por transformar el problema aproximado para el caso en que se haya utilizado la linealización en forma δ. Para este caso el problema de optimización aproximado se podía formular de la siguiente manera:

$$\begin{aligned} &\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta f_{kj})\delta_{kj} - \sum_{j=1}^{n} f_{0j} \\ &\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta g_{kij})\delta_{kj} \{\leq, =, \geq\} b_i - \sum_{j=1}^{n} g_{0ij} \quad \text{para } i = 1, \cdots, m \\ &\quad 0 \leq \delta_{kj} \leq 1 \quad \text{para todo k, j} \end{aligned} \qquad (8.62)$$

donde además se requiere que si $\delta_{kj} > 0$ entonces $\delta_{uj} = 1$ para u = 1, ... , k-1.

Para convertir dicho problema a un problema de programación entera mixta lo que debe hacerse es representar la condición: "si $\delta_{kj} > 0$ entonces $\delta_{uj} = 1$ para u = 1, ... , k-1" utilizando variables enteras. Obsérvese que la condición anterior es equivalente a decir: $\delta_{(k+1)j} = 0$ excepto $\delta_{kj} = 0$. Para representar estas restricciones, se introducirán un conjunto de variables ψ_{kj} que sólo podrán tomar valores enteros 0 ó 1. Supongamos que para cada k y j, se introducen las siguientes restricciones:

$$\begin{aligned} \delta_{kj} &\geq \psi_{kj} \\ \delta_{(k+1)j} &\leq \psi_{kj} \end{aligned} \qquad (8.63)$$

Entonces si $\psi_{kj} = 1$, de las restricciones anteriores se deduce que $\delta_{kj} \geq 1$, o sea, $\delta_{kj} = 1$ y que $\delta_{(k+1)j} \leq 1$, o sea, no se aplica ninguna restricción sobre $\delta_{(k+1)j}$.

Por otro lado, si $\psi_{kj} = 0$, entonces se requiere que $\delta_{kj} \geq 0$, es decir, no se aplica ninguna restricción sobre δ_{kj}, mientras que $\delta_{(k+1)j} \leq 0$, o sea, $\delta_{(k+1)j} = 0$.

Se observa por lo tanto, que mediante estas dos nuevas restricciones y la variable entera ψ_{kj} que se ha introducido se consigue lo que se deseaba, traducir la condición que acompañaba al problema aproximado al formato de un problema entero.

Así mismo, las restricciones anteriores garantizan que si $\psi_{kj} = 1$, entonces se debe cumplir que $\psi_{uj} = 1$ para u = 1,..., k-1. Se puede comprobar fácilmente que si se toma $\psi_{uj} = 1$ y $\psi_{(u-1)j} = 0$, como ejemplo de un caso que no cumple la anterior afirmación, se llega a la contradicción de que $\delta_{uj} \leq 0$ y $\delta_{uj} \geq 1$.

Para garantizar que las variables enteras ψ_{kj} toman sólo valores 0 ó 1, sólo hace falta imponer la siguiente restricción adicional $\psi \geq 0$ para todo k y j. No es necesario imponer las condiciones $0 \leq \psi_{kj} \leq 1$ porque las restricciones $\delta_k \geq \psi_{uj}$ y $0 \leq \delta_{kj} \leq 1$ aseguran que ψ_{kj} no pueda tomar valores mayores que la unidad.

Teniendo en cuenta todo lo comentado anteriormente el problema separable linealizado se puede formular como el siguiente problema de programación entera mixta:

$$\begin{aligned} &\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta f_{kj})\delta_{kj} - \sum_{j=1}^{n} f_{0j} \\ &\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} (\Delta g_{kij})\delta_{kj} \{\leq, =, \geq\} b_i - \sum_{j=1}^{n} g_{0ij} \quad \text{para } i = 1, \cdots, m \\ &\qquad 0 \leq \delta_{kj} \leq 1 \quad \text{para todo k, j} \\ &\qquad \delta_{kj} - \psi_{kj} \geq 0 \quad \text{para todo k, j} \\ &\qquad \delta_{(k+1)j} - \psi_{kj} \leq 0 \quad \text{para todo k, j} \\ &\qquad \psi_{kj} \geq 0 \ \text{con } \psi_{kj} \text{ entero, para todo k, j} \end{aligned} \qquad (8.64)$$

B. Forma λ

De forma análoga el problema de optimización aproximado mediante la forma λ puede también formularse como un problema de programación entera mixta, con la finalidad de poder garantizar que el óptimo que se va obtener sea global.

Tal como vimos al inicio del capítulo un problema no lineal separable se podía aproximar utilizando la forma λ de la siguiente manera:

$$\max\{\min\} \sum_{j=1}^{n} \sum_{k=0}^{r_j} f_{kj}\lambda_{kj}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} \sum_{k=0}^{r_j} g_{kij}\lambda_{kj} \{\leq, =, \geq\} b_i \quad \text{para } i = 1, \cdots, m \tag{8.65}$$

$$\sum_{k=0}^{r_j} \lambda_{kj} = 1 \quad \text{para } j = 1, \cdots, n$$

$$\lambda_{kj} \geq 0 \quad \text{para todo } k, j$$

debiéndose cumplir que para cualquier j, no más de dos λ_{kj} pueden ser positivos (diferentes de cero) y además los dos que sean diferentes de cero deben ser adyacentes.

La formulación de dicho problema como un problema de programación entera mixta, implica traducir adecuadamente, utilizando variables enteras, las restricciones sobre que λ_{kj} pueden ser diferentes de cero. Para ello introduciremos un conjunto de variables enteras ψ_{kj} que sólo puedan tomar valores 0 ó 1. A continuación impondremos la siguiente condición:

$$\sum_{k} \psi_{kj} = 1 \text{ para } j = 1, \cdots, n \tag{8.66}$$

- **Relación entre el óptimo del problema original y aproximado**

Si la longitud de la rejilla utilizada para aproximar a linealmente a trozos las funciones no lineales del problema original se escoge lo suficientemente pequeña, el valor del óptimo del problema original se puede aproximar tanto como se desee al valor del óptimo del problema aproximado. Para probar esta afirmación, presentamos los siguientes teoremas cuya demostración puede encontrarse en "Nonlinear Programming" de Bazaraa [Bazaraa93].

Teorema:

Consideremos los problemas original P y aproximado AP, respectivamente, tal como se han enunciado en las primeras secciones de este capítulo. Para $j \notin L$, supongamos que f_j y g_{ij} para i = 1,...,m son convexas, y además, sean $\hat{f}_j$ y $\hat{g}_{ij}$ sus aproximaciones lineales a trozos en el intervalo $[a_j, b_j]$. Para $j \notin L$ y para i = 1,...,m, sea c_{ij} tal que $\left|g'_{ij}(x_j)\right| \leq c_j$ para $x_j \in [a_j, b_j]$. Además, para $j \notin L$ sea c_j tal que $\left|f'_j(x_j)\right| \leq c_j$ para $x_j \in [a_j, b_j]$. Para $j \notin L$, sea δ_j la máxima longitud de rejilla utilizable para la variable x_j. Entonces se cumple que:

$$\hat{f}(x) \geq f(x) \geq \hat{f}(x) - c$$

$$\hat{g}_i(x) \geq g_i(x) \geq \hat{g}_i(x) - c \quad \text{para } i = 1, \cdots, m$$

donde:

$$c = \max_{0 \leq i \leq m} \{\bar{c}_i\}$$

con

$$\bar{c}_0 = \sum_{j \notin L} 2c_j\delta_j$$

$$\bar{c}_i = \sum_{j \notin L} 2c_{ij}\delta_j \quad \text{para } i = 1, \cdots, m$$

Teorema:

Consideremos el problema original P, tal como se ha definido en la primera sección de este capítulo. Sea $L=\{j: f_j, g_{ij} \quad \text{para } i=1,\cdots,m \quad \text{son lineales}\}$. Para $j\notin L$, sean $\hat{f}_j$ y $\hat{g}_{ij}$ las aproximaciones lineales a trozos de f_j y g_{ij}, para i = 1, ..., m respectivamente. Sea el problema aproximado AP al problema original, y sea el problema aproximado lineal a trozos LAP, supongamos que f_j y g_{ij} para i = 1,...,m son convexas. Sea $\bar{x}$ una solución óptima al problema P. Sea $\hat{x}_j$ para $j\in L$ y $\hat{\lambda}_{vj}$ para v = 1,...,k_j una solución óptima para el problema LAP tal que el vector $\hat{x}$, cuyas componentes $\hat{x}_j$ para $j\in L$ y $\hat{x}_j=\sum_{v=1}^{k_j}\hat{\lambda}_{vj}x_{vj}$ para $j\notin L$, es una solución óptima para el problema AP. Sean $\hat{u}_i \geq 0$ los multiplicadores de Lagrange óptimos asociados con la restricción $\hat{g}_i(x)\leq 0$ para i = 1, ..., m. Entonces se cumple que:

(a) $\hat{x}$ es una solución factible del problema P.

(b) $0\leq f(\hat{x})-f(\bar{x})\leq c(1+\sum_{i=1}^{m}\hat{u}_i)$ donde *c* se calcula de la forma indicada en el teorema anterior.

Los multiplicadores de Lagrange $\hat{u}_i$ para i = 1,...,m a los que hace referencia el teorema anterior están disponibles de forma automática a partir del tableau simplex óptimo del problema LAP. De forma que. cuando se resuelve el problema aproximado, se puede utilizar dicho teorema para determinar la máxima desviación

$$f(\hat{x})-f(\bar{x})\leq c(1+\sum_{i=1}^{m}\hat{u}_i) \qquad (8.67)$$

del valor real de la función objetivo del problema original. Debe observarse que al reducirse la longitud del paso de la rejilla utilizada para aproximar linealmente a trozos las funciones f_j y g_{ij}, *c* se hará menor, y por lo tanto, se obtendrá una mejor aproximación del óptimo real.

- **Generación de los puntos de la rejilla**

La precisión de este método de optimización depende en gran medida del número de puntos de la rejilla utilizada para cada variable. Sin embargo, a medida que el número de puntos de la rejilla aumenta, el número de variables en el problema lineal aproximado LAP también aumenta. Una alternativa consistiría en utilizar inicialmente una rejilla poco fina para utilizar más adelante una rejilla más fina alrededor de la solución óptima obtenida con la rejilla gruesa. Una alternativa atractiva consiste en generar los puntos de la rejilla a medida que resulte necesario. Este enfoque es el que se discutirá a continuación.

Consideremos el problema de optimización aproximado linealmente a trozos LAP. Sea x_{vj} para $v=1,\cdots,k_j$ y $j\notin L$ la rejilla de puntos considerada hasta el momento. Sea $\hat{x}_j$ para $j\in L$ y $\hat{\lambda}_{vj}$ para $v=1,\cdots,k_j$ y $j\notin L$ la solución del problema lineal aproximado LAP. Además, sea $\hat{u}_i \geq 0$ para $i=1,\cdots,m$ los multiplicadores de Lagrange óptimos asociados con las *m* primeras restricciones , y sea $\hat{v}_j$ para $j\notin L$ el multiplicador de Lagrange asociado con la restricción $\sum_{v=1}^{k_j}\lambda_{vj}=1$. Obsérvese que la solución $\hat{x}_j$, $\hat{\lambda}_{vj}$, $\hat{u}_i$ y $\hat{v}_j$ satisface las condiciones de Kuhn-Tucker para el problema lineal aproximado.

El objetivo es determinar si se debe considerar una rejilla de puntos adicional para cualquiera de las variables x_j para $j\notin L$ a fin de conseguir una mejor aproximación lineal a trozos en el sentido que, si al considerarla en la definición del problema lineal aproximado, el valor mínimo de la función objetivo

será menor. Para algún $j \notin L$, supongamos que consideráramos un punto de la rejilla $x_{\gamma j}$. Se puede verificar fácilmente que

$$f_j(x_{\gamma j}) + \sum_{i=1}^{m} \hat{u}_i g_{ij}(x_{\gamma j}) + \hat{v}_j \geq 0 \qquad (8.68)$$

entonces, si hacemos $\hat{\lambda}_{\gamma j} = 0$ se cumplirán las condiciones de Kuhn-Tucker para el problema lineal revisado. Sin embargo, puesto que no sabemos donde se encuentra este nuevo punto de la rejilla, podremos responder a la pregunta de si todos los x_j tales que $a_j \leq x_j \leq b_j$ para $j \notin L$ satisfacen la condición anterior resolviendo los siguientes subproblemas para cada $j \notin L$

$$\begin{aligned} \min \quad & f_j(x_{\gamma j}) + \sum_{i=1}^{m} \hat{u}_i g_{ij}(x_{\gamma j}) + \hat{v}_j \\ \text{s.t.} \quad & a_j \leq x_j \leq b_j \end{aligned} \qquad (8.69)$$

Si el valor mínimo de la función objetivo es no negativo para todo $j \notin L$, entonces no podemos encontrar una nueva rejilla de puntos que contradiga a $f_j(x_{\gamma j}) + \sum_{i=1}^{m} \hat{u}_i g_{ij}(x_{\gamma j}) + \hat{v}_j \geq 0$.

El teorema que presentaremos a continuación permite asegurar, si es el caso, que la solución actual es la óptima del problema original P y si el valor mínimo de la función objetivo es negativo para algún $j \notin L$, se pude obtener una mejor aproximación para el problema original P. Además, el teorema proporciona las cotas sobre el valor óptimo la función objetivo del problema P a cada iteración.

Teorema:

Consideremos el problema original P tal como se ha definido en la primera sección de este capítulo. Sea $L = \{j : f_j, g_{ij} \text{ para } i = 1, \cdots, m \text{ son lineales}\}$. Supongamos, sin pérdida de generalidad, que $f_j(x_j)$ es de la forma $c_j x_j$ y $g_{ij}(x_j)$ es de la forma $a_{ij} x_j$ para i = 1, ... , m y para $j \in L$. Utilizando la rejilla de puntos x_{vj} para v = 1,... , k_j para $j \notin L$, sea el problema lineal aproximado LAP. Para $j \notin L$, supongamos que f_j y g_{ij}, son convexas para i = 1, ..., m. Sea $\hat{x}_j$ para $j \in L$ y $\hat{\lambda}_{vj}$ para v = 1, ... ,k_j y $j \notin L$ el óptimo del problema lineal aproximado LAP con un valor de la función objetivo asociado $\hat{z}$. Sean $\hat{u}_i \geq 0$ para i = 1, ... , m los multiplicadores de Lagrange asociados con las restricciones $\sum_{v=1}^{k_j} \lambda_{vj} = 1$ del problema lineal aproximado LAP. Para cada $j \notin L$, consideremos el siguiente problema:

$$\begin{aligned} \min \quad & f_j(x_{\gamma j}) + \sum_{i=1}^{m} \hat{u}_i g_{ij}(x_{\gamma j}) + \hat{v}_j \\ \text{s.t.} \quad & a_j \leq x_j \leq b_j \end{aligned}$$

donde $[a_j, b_j]$ con $a_j, b_j \geq 0$ es el intervalo de interés para x_j. Sea $\bar{z}_j$ el valor óptimo de la función objetivo del problema de optimización anterior. Entonces, se cumple que:

(a) $\sum_{j \notin L} \bar{z}_j - \sum_{i=1}^{m} \hat{u}_i p_i \leq \sum_{j=1}^{n} f_j(\bar{x}_j) \leq \sum_{j=1}^{n} f_j(\hat{x}_j) \leq \hat{z}$ donde $\hat{x}_j = \sum_{v=1}^{k_j} \hat{\lambda}_{vj} x_{vj}$ para $j \notin L$ y $\bar{x} = (\bar{x}_1, \cdots, \bar{x}_n)^t$ es una solución óptima del problema original P.

(b) Si $\bar{z}_j + \hat{v}_j \geq 0$ para todo $j \notin L$, entonces $\hat{x} = (\hat{x}_1, \cdots, \hat{x}_n)^t$ es una solución óptima del problema original P. Además, $\sum_{j=1}^{n} f_j(\hat{x}_j) = \hat{z}$.

(c) Si $\bar{z}_j + \hat{v}_j < 0$ para algún $j \notin L$, sea x_{vj} la solución óptima que condujo a $\bar{z}_j < -\hat{v}_j$, entonces añadiendo el punto x_{vj} a la rejilla que define el problema lineal aproximado LAP, se obtendrá un nuevo problema lineal aproximado LAP con una valor para la función objetivo que no será mayor que $\hat{z}$.

Utilizando como base los resultados presentados en este teorema, vamos a describir un procedimiento para la generación de la rejilla de puntos necesaria para obtener un problema lineal aproximado LAP que nos permita resolver un problema de la forma

$$\begin{aligned} &\min \sum_{j=1}^{n} f_j(x_j) \\ &\text{s.t.} \quad \sum_{j=1}^{n} g_{ij}(x_j) \le 0 \quad \text{para } i = 1, \cdots, m \\ &\qquad x_j \ge 0 \quad \text{para } j = 1, \cdots, n \end{aligned} \tag{8.70}$$

Sea $L = \{j : f_j, g_{ij} \quad \text{para } i = 1, \cdots, m \quad \text{son lineales}\}$. El procedimiento que presentaremos a continuación permitirá obtener una solución óptima para el problema de optimización anterior utilizando el método simple sin base de entrada restringida si g_{ij}es convexa para i = 1,...,m y $j \notin L$, y f_j es estrictamente convexa para $j \notin L$.

Paso 0. Inicialización. Definiremos $a_j, b_j \ge 0$ de forma que todos los puntos factibles satisfagan $x_j \in [a_j, b_j]$ para $j \notin L$. Para cada $j \notin L$, seleccionaremos una rejilla de puntos. Sea k_j igual al número de puntos de la rejilla para $j \notin L$.

Paso 1. Resolución del problema lineal aproximado LAP. Sea la solución óptima $\hat{x}_j$ para $j \in L$ y $\hat{\lambda}_{vj}$ para $v = 1, \ldots, k_j$ y $j \notin L$. Sean $\hat{u}_i$ los multiplicadores de Lagrange asociados con las primeras m restricciones, y sea $\hat{v}_j$ para $j \notin L$ el multiplicador de Lagrange asociado con $\sum_{v=1}^{k_j} \lambda_{vj} = 1$.

Paso 2. Resolución del problema de optimización

$$\begin{aligned} &\min \quad f_j(x_{vj}) + \sum_{i=1}^{m} \hat{u}_i g_{ij}(x_{vj}) + \hat{v}_j \\ &\text{s.t.} \quad a_j \le x_j \le b_j \end{aligned}$$

para $j \notin L$. Sea $\bar{z}_j$ para $j \notin L$, el valor óptimo de la función objetivo. Si $\bar{z}_j + \hat{v}_j \ge 0$ para todo $j \notin L$, entonces el algoritmo finaliza, siendo la solución óptima del problema original P $\hat{\mathbf{x}}$, cuyas componentes son $\hat{x}_j$ para $j \in L$ y $\hat{x}_j = \sum_{v=1}^{k_j} \hat{\lambda}_{vj} x_{vj}$. En caso contrario, saltaremos al paso 3.

Paso 3. Sea $\bar{z}_p + \hat{v}_p = \min_{j \notin L} (\bar{z}_j + \hat{v}_j) < 0$. Sea x_{vp} la solución óptima que conduce a $\bar{z}_j < -\hat{v}_j$. Sea $v = k_p + 1$, entonces reemplazamos k_p por k_{p+1} y saltamos al paso 1.

8.7 Métodos Computacionales de Programación Entera Mixta [Fuente95]

A continuación presentaremos algunos de los métodos computacionales para resolver un problema de programación entera mixta. Un **problema de optimización entero mixto** es un problema con la siguiente estructura

$$\begin{aligned} \max \quad & (c^T x + h^T y) \\ \text{s.t.} \quad & Ax + Gy \le b \\ & x \ge 0, y \ge 0 \\ & x \in Z^n \\ & y \in R^p \end{aligned} \qquad (8.71)$$

Para este problema de optimización la región factible se define como el conjunto

$$S = \{x \in Z_+^n, y \in R_+^p : Ax + Gy \le b\} \qquad (8.72)$$

Un punto $[x^T, y^T]^T \in S$ se denomina factible. Al punto $[x^{*T}, y^{*T}]^T \in S$, tal que

$$c^T x^* + b^T y^* \ge c^T x + h^T y \quad \text{para} \quad [x^T, y^T] \in S \qquad (8.73)$$

se le denomina solución óptima del problema.

La técnicas básicas de resolución de este tipo de problemas se bajan en el concepto de la **relajación**. La idea básica de la relajación es sustituir el problema original por otro más fácil de resolver en el que se relajan alguna o algunas de las condiciones, obteniéndose de su resolución una cota de la función objetivo del programa original. Mediante un proceso iterativo que integre esta relajaciones y las resolución de los subproblemas, se va acotando cada vez más la función objetivo deseada hasta que se llega a la solución final.

Una relajación del programa entero PE

$$\max \{c^T x : x \in S\} \qquad (8.74)$$

la constituye cualquier subproblema PR de la forma

$$\max \{z_{PR}(x) : x \in S_{PR}\} \qquad (8.75)$$

con las propiedades siguientes:

$$S_{PR} \supseteq S \qquad (8.76)$$

y

$$c^T x \le z_{PR}(x) \quad \text{para } x \in S \qquad (8.77)$$

Las propiedades anteriores se escogen con el siguiente criterio:

"si el problema (PR) no es factible, tampoco lo es (PE). Si (PR) sí es factible, los valores de las funciones objetivo cumplen que $z_{PE} \le z_{PR}$*".*

La relajación más evidente que se puede utilizar de un programa entero resulta de cancelar la restricción de que las variables han de ser enteras. Si el conjunto o región factible S, del programa entero se define

$$S = P \cap Z^n \qquad (8.78)$$

la relajación lineal es

$$\max \{c^T x : x \in P\} \qquad (8.79)$$

donde $P = \{x \in R^n : Ax \leq b, x \geq 0\}$.

Los dos grupos de algoritmos que se presentaremos a continuación están siguen un esquema general de algoritmo basado en relajaciones sucesivas del problema original. La estructura de dicho algoritmo es el siguiente:

Algoritmo de Relajaciones Sucesivas

Paso 0. *Inicialización.* Hacer i = 1, $w^* = \infty$ y $z^* = -\infty$. Escoger un $S_R^{(1)} \supseteq S$ tal que $z_R^{(1)} \geq c^T x$ para $x \in S$.

Paso 1. *Relajación.* Resolver el problema PE relajado $PR^{(i)}$

$$\max \{z_R^{(i)}(x) : x \in S_R^{(i)}\}$$

Paso 2. *Comprobación de Óptimo.* Sea $x^{(i)}$ la solución del problema anterior, $R^{(i)}$. Si $x^{(i)} \in S$ y $z_R^{(i)} = c^T x^{(i)}$, ésta es la solución óptima de PE. La función objetivo óptima es $w^* = c^T x^{(i)} = z^*$.

Paso 3. *Mejora de la solución.* Hacer $w^* = z_R^{(i)}$, $z^* = c^T x^{(i)}$ si $x^{(i)} \in S$ y $i \leftarrow i+1$. Escoger S_R^{i+1} tal que $S \subseteq S_R^{i+1} \subseteq S_R^i$ y $z_R^{i+1}(x)$ tal que $c^T x \leq z_R^{i+1}(x) \leq z_R^i(x)$ para $x \in S$ con $S_R^{i+1} \neq S_R^i$ o $z_R^{i+1}(x) \neq z_R^i(x)$. Ir al paso 1.

- **Algoritmo de los Planos Cortantes de Gomory**

El problema que resuelve dicho algoritmo es

$$\max \{c^T x : x \in S^e\} \qquad (8.80)$$

donde

$$S^e = \{x \in Z^n : Ax = b, x \geq 0\} \qquad (8.81)$$

esta escrito de la siguiente forma

$$\max \{x_0 : [x_0, x^T]^T \in S^0\} \qquad (8.82)$$

donde

$$S^0 = \{x_0 \in Z^1, x \in Z_+^n : x_0 - c^T x = 0, Ax = b\} \qquad (8.83)$$

Supondremos que se dispone de una solución y base óptimas de la relajación lineal del programa entero. El problema se puede escribir entonces

$$\begin{aligned} &\max \quad x_0 \\ &\text{s.t.} \quad x_{B_i} + \sum_{j \in H} \bar{a}_{ij} = \bar{a}_{i0} \quad \text{para } i = 0,1,\cdots,m \\ &x_{B_o} \in Z_+^1 \quad \text{para } i = 1,\cdots,m \\ &x_j \in Z^1, x_{B_i} \in Z_+^1 \quad \text{para } i = 1,\cdots,m \\ &x_j \in Z_+^1 \quad \text{para } j \in H \end{aligned} \qquad (8.84)$$

donde $x_0 = x_{B_o}$, x_{B_i}, i=1,...,m, son las variables básicas y x_j, $j \in H \subset N = \{1,\cdots,n\}$, las no básicas. Como esta base es factible en el programa primal y dual se cumple que $\bar{a}_{i0} \geq 0$ para $i = 1,\ldots,m$ y que $\bar{a}_{i0} \geq 0$ para $j \in H$.

Suponiendo que en el problema anterior existe un i tal que $\bar{a}_{i0} \notin Z^1$ se tiene la siguiente proposición que define el **corte fraccionario de Gomory**:

"si $\bar{a}_{i0} \notin Z^1$, *se tiene que*

$$\sum_{j \in H} f_{ij} x_j = f_{i0} + x_{n+1} \quad x_{n+1} \in Z_+^1 \qquad (8.85)$$

donde $f_{ij} = \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$ *para* $j \in H$ *y* $f_{i0} = \bar{a}_{i0} - \lfloor \bar{a}_{i0} \rfloor$ *es una desigualdad válida en* S^0*"*.

El algoritmo completo para resolver programas enteros puros (no hay variables reales) basándose en los cortes de Gomory es el se presenta a continuación.

Algoritmo de los Planos Cortantes de Gomory

Paso 0. *Inicialización*. Hacer $i = 1$, $z_R^{(1)} = x_0$ y $S_R^{(1)} = \{x_0 \in R, x \in R_+^n : x_o - c^T x = 0, Ax = b\}$

Paso 1. *Resolución de la Relajación Lineal*. Resolver el problema PE relajado $PR^{(i)}$

$$\max \{x_0 : [x_0, x^T]^T \in S_R^{(i)}\}$$

Si PR^i es factible y tiene solución óptima, supongamos que es $[x_0^{(i)}, (x^{(i)})^T]^T$, continuar.

Paso 2. *Comprobación de Óptimo*. Si $x^{(i)} \in Z_+^n$, ésta es la solución óptima del programa entero.

Paso 3. *Comprobación de no factibilidad*. Si $PR^{(i)}$ no es factible, el programa entero original no es factible.

Paso 4. *Adición de un corte de Gomory*. Escoger una fila $x_{B_i} + \sum_{j \in H^{(i)}} \bar{a}_{ij}^{(i)} x_j = \bar{a}_{i0}^{(i)}$ con $\bar{a}_{i0}^{(i)} \notin Z^1$. Hacer

$$\sum_{j \in H^{(i)}} f_{ij} x_j - x_{n+i} = f_{i0}, \quad x_{n+i} \in Z_+^1$$

el corte de Gomory de esa fila. Hacer

$$S_R^{(i+1)} = S_R^{(i)} \cap \left\{ x_0 \in R, x \in R_+^{n+i} : \sum_{j \in H^{(i)}} f_{ij} x_j - x_{n+i} = f_{i0} \right\}$$

Paso 5. Hacer $i \leftarrow i+1$ e ir al paso 1.

- **Algoritmo de Branch and Bound**

Este algoritmo se inscribe dentro de la familia de los denominados algoritmos enumerativos. El conjunto $\{S^i : i = 1, \ldots, k\}$ se denomina **división de la región factible** del programa entero mixto, S si

$$S = \bigcup_{i=1}^{k} S^i \qquad (8.86)$$

Una división se denomina **partición** si $S^i \cap S^j = \varnothing$ para i,j =1,...,k con i≠j. Sea el problema de optimización entero mixto PE^i

$$\max \quad \{c^T x + h^T y : x \in S^i\} \qquad (8.87)$$

donde $\{S^i\}_{i=1}^k$ es una división de S, entonces se cumple que la solución al problema entero mixto z_{PEM} se determinará mediante

$$z_{PEM} = \max_{i=1,\ldots,k} \{z_{PE}^i\} \qquad (8.88)$$

La división de la región factible se suele hacer de forma recursiva. La forma más simple de hacer dicha división es la indica en la Fig. 8.2. En ella

$$S^{\delta_1 \cdots \delta_k} = S \cap \{x \in B^n : x_j = \delta_j \in \{0,1\}, \text{para } j = 1,\ldots,k\} \qquad (8.89)$$

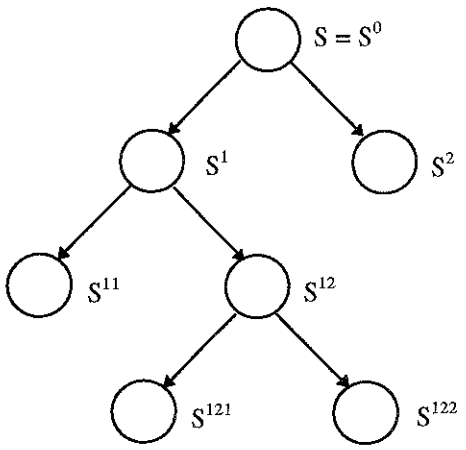


Fig. 8.2 División recursiva de una región factible

Esta división así obtenida es una partición de S. Llevada a un extremo, la división del conjunto factible contempla la enumeración de todos los elementos de S. En problemas prácticos la enumeración y análisis de todos los elementos del conjunto factible es inviable cuando el número de variables es grande. Por lo tanto, al implementar prácticamente dichos métodos se debe contemplar alguna forma de eliminar ramas en el árbol de búsqueda que se forma, al que en lo sucesivo denominaremos **árbol enumerativo**.

Si la región factible S se divide en subconjuntos $\{S^1,\cdots,S^k\}$, y se puede establecer de alguna manera que no es necesario a partir de un determinado momento seguir dividiendo un subconjunto S^i, se dice entonces que el árbol enumerativo puede podarse a partir del nudo correspondiente.

El árbol enumerativo se puede podar a partir del nudo correspondiente a S^i si se cumplen cualquiera de las siguientes condiciones:

(a) *No factibilidad*: $S^i = \varnothing$
(b) *Optimalidad*: se ha llegado al óptimo de PEM^i
(c) *Existencia de un valor dominante*: $z_{PEM}^i \leq z_{PEM}$
Para no tener que resolver el problema de programación entero mixto PEM^i se pueden utilizar diversos procedimientos. El más habitual: la relajación de programación de PEM^i, o sea, PL^i con $S^i \subseteq S_{PL}^i$ y $z_{PL}^i \geq c^T x + h^T y$ para $x \in S^i$.

Por lo tanto:

El árbol enumerativo puede podarse a partir del nudo correspondiente a un conjunto S^i si se cumple cualquiera de las siguientes condiciones:

(a) *No factibilidad*: el programa PL^i no es factible
(b) *Optimalidad*: la solución óptima (x_{PL}^i, y_{PL}^i) de PL^i cumple que $x_{PL}^i \in S^i$ y que $z_{PL}^i = c^T x_{PL}^i + h^T y_{PL}^i$

(c) *Existencia de un valor dominante*: $z^i_{PL} \le \underline{z}_{PE}$, donde $\underline{z}_{PE}$ es el valor de alguna solución factible de PEM.

El algoritmo de Branch and Bound aplicado a la resolución de un problema de programación entera mixta tiene la siguiente estructura:

Algoritmo de Branch and Bound

Paso 0. *Inicialización*. Hacer L={PEM}, S^o=S, $\overline{z}^o = \infty$ y $\underline{z}_{PEM} = -\infty$.

Paso 1. *Comprobación de final*. Si L=∅, la solución (x^o,y^o) con función objetivo

$$\underline{z}_{PEM} = c^T x^o + h^T z^o$$

es óptima.

Paso 2. *Selección del problema, relajación lineal y resolución*. Seleccionar y borrar de L un problema PEM^i. Resolver su relajación lineal PL^i. Sea z^i_{PL} el valor óptimo de la función objetivo de este problema y (x^i_{PL}, y^i_{PL}) su solución óptima si existe.

Paso 3. *Poda*. Si $z^i_{PL} \le \underline{z}_{PEM}$ saltaremos al paso 1, mientras que si $x^i_{PL} \notin S^i$ saltaremos al paso 4, y finalmente si $x^i_{PL} \in S^i$ y $c^T x^i_{PL} + h^T y^i_{PL} > \underline{z}_{PEM}$, hacer $\underline{z}_{PEM} = c^T x^i_{PL} + h^T y^i_{PL}$. Borrar de L todos los subproblemas tales que $\overline{z}^i \le \underline{z}_{PE}$. Si $c^T x^i_{PL} + h^T y^i_{PL} = z^i_{PL}$ saltaremos al paso 1, si no, entonces saltar al paso 4.

Paso 4. *División*. Sea $\left\{S^{ij}\right\}_{j=1}^{k}$ una división de S. Añadir los problemas $\left\{PEM^{ij}\right\}_{j=1}^{k}$ a L, donde $\overline{z}^{ij} = z^i_{PL}$ para j=1,...,k y saltar al paso 1.

Analicemos con un poco más detalle el funcionamiento de este algoritmo.

A. Criterios de eliminación (poda)

Al resolver las relajaciones lineales, los criterios de rechazo presentados anteriormente: no factibilidad, optimalidad y valor dominante son aplicables directamente. Supóngase que la relajación lineal en el nudo *i* del árbol enumerativo PL^i es

$$\max \quad \{c^T x + h^T y : x \in S^i_{PL}\} \qquad (8.90)$$

donde

$$S^i_{PL} = \{x \ge 0 : A^i x \le b^i\} \qquad (8.91)$$

Si el programa lineal PL^i tiene solución óptima, y la designamos mediante x^i, entonces las condiciones para eliminar el nudo *i*, y por lo tanto sus nodos hijos son:

(a) *No factibilidad*: el programa PL^i no es factible, o sea, $S^i_{PL} = \emptyset$.

(b) *Optimalidad*: $x^i \in Z^n_+$

(c) *Existencia de un valor dominante*: $z^i_{PL} \le \underline{z}_{PE}$, donde $\underline{z}_{PE}$ es el mejor valor de las soluciones factibles encontradas hasta el momento.

B. División

La división se debe hacer añadiendo condiciones lineales, debido a que se usa relajación lineal en cada nudo del árbol enumerativo. Una forma evidente de hacer esto es tomar

$$S = S^1 \cup S^2 \qquad (8.92)$$

con

$S^1 = S \cap \{x \ge 0 : d^T x \le d_o\}$ y $S^2 = S \cap \{x \ge 0 : d^T x \ge d_o + 1\}$, donde $[d^T, d_o] \in Z^{n+1}$. Si x^o es la solución del programa lineal inicial PL^o

$$\max \{c^T x + h^T y: x \geq 0, Ax + Gy \leq b\} \qquad (8.93)$$

entonces el vector $[d^T, d^\circ]^T$ se puede escoger de manera que $d_o < d^T x^\circ < d_o + 1$. Proceder de esta forma es lo más recomendable pues se obtiene un $x^\circ \notin S^1_{PL} \cup S^2_{PL}$, lo que permite que para i=1,2,

$$z^i_{PL} = \max\{c^T x + h^T y; x \in S^i_{PL}\} < z^\circ_{PL} \qquad (8.94)$$

En la práctica sólo se usan vectores $[d^T, d^\circ]^T$ muy concretos:

(a) *Dicotomías de variables.* En este caso $d = e_j$ para algún $j \in N$. El punto x° será no factible en las relajaciones resultantes si $x^\circ_j \notin Z^1$ y $d_o = \lfloor x_j \rfloor$, tal como se muestra en la Fig. 8.3. Nótese que si $x_j \in B^1$, la rama izquierda hace $x_j = 0$ y la derecha $x_j = 1$. Una importante ventaja práctica de esta división es que las restricciones lineales que se añaden a la relajación lineal correspondiente son simples cotas inferiores o superiores. Sólo será necesario en la optimización del nuevo programa lineal tener en cuenta esas nuevas cotas y resolverlo mediante el método dual del símplex, de forma que el tamaño de la base será el mismo.

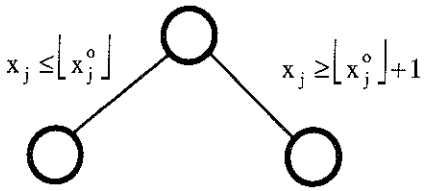


Fig. 8.3 División en un árbol enumerativo por dicotomía de la variable x_j

(b) *Dicotomías de cotas superiores generalizadas.* Si el problema tuviese cotas superiores generalizadas,

$$\sum_{j \in Q} x_j = 1 \qquad (8.95)$$

para algún $Q \subseteq N$, la división a llevar a cabo sería la de la Fig. 8.4. Obsérvese que x° será no factible en la relajación resultante si

$$0 < \sum_{j \in Q_1} x^\circ_j < 1 \qquad (8.96)$$

donde Q_1, es un subconjunto no vacío de Q.

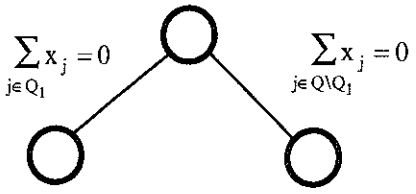


Fig. 8.4 Dicotomía debida a la existencia de cotas superiores generalizadas

(c) Si una variable x_j está acotada, $l_j \leq x_j \leq u_j$, otra forma de proceder sería considerar cada valor entero posible de x_j por separado, tal como se indica en la Fig. 8.5. Como se puede comprender, si el número de variables es un poco elevado, pudiendo tomar cada una varios valores enteros, el problema puede adquirir un tamaño gigantesco. Ningún solver de programación entera mixta utiliza dicha posibilidad.

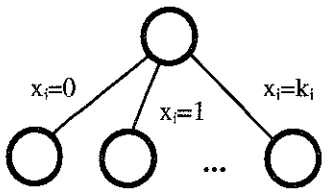


Fig. 8.5 División del árbol enumerativo en tantas ramas como valores enteros puede tomar la variable entera x_j

Debe observarse que cada una de las particiones expuestas es una partición. En lo que sigue supondremos que la división se realiza por dicotomía de variables.

C. Selección del nudo a estudiar

Una de las operaciones de más trascendencia del algoritmo de branch and bound queda reflejada en el paso 2 de éste: dada una lista L de subproblemas activos o, dicho de otra forma, de subárboles del árbol enumerativo con nudos no rechazados, qué nudo elegir como el próximo a examinar. Las opciones son dos:

(a) analizar uno siguiendo una reglas preestablecidas
(b) escoger uno de acuerdo con la información disponible en ese momento sobre las cotas de las variables, número de nudos activos, etc.

La primera opción la podríamos encuadrar dentro de un apartado denominado de reglas apriorísticas mientras que la segunda opción en otro de reglas adaptativas.

Existen multitud de reglas apriorísticas que se pueden utilizar. La más extendida es la denomina LIFO ("Last In First Out"), que consiste en analizar primero todos los nudos hijos de un determinado nudo y luego volver hacia atrás escogiendo otro del mismo nivel del nudo padre que no haya sido todavía analizado. Esta regla, junto con otra que determine qué variable va a ser la de separación siguiente y con cuál de las dos ramas a que esta separación da lugar empezar, permiten analizar el problema en su totalidad. La regla LIFO tiene dos ventajas fundamentales:

(a) La relajación lineal de un nudo hijo se obtiene de la lineal del nudo padre sin más que añadir una restricción simple de cota en una variable. A partir de la solución óptima del programa lineal que define el nudo padre, utilizando el método dual del simple, sin reinvertir la base ni modificar las estructuras de datos substancialmente, se obtiene la que define el nudo hijo.
(b) La experiencia dice que las soluciones factibles se encuentran más bien en puntos profundos del árbol enumerativo.

Otra regla apriorística que se puede utilizar es la denominada búsqueda en amplitud o anchura. Consiste en analizar todos los nudos de un determinado nivel antes de pasar al otro más profundo. Aunque esta variante no obtiene resultados prácticos relevantes comparándolos con los de la anterior, sí es muy usada en códigos especializados en la resolución de programas especiales en variables binarias que utilizan técnicas heurísticas.

Cualquiera que sea la regla, los criterios más razonables que deben considerarse para seleccionar un nudo a analizar son:

(a) Escoger aquel nudo que se ha de estudiar en cualquier caso. Si sólo existe un nudo con la cota más alta, estudiarlo. Es decir, cuando un nudo se rechaza, escoger aquel de los todavía no analizados que tenga la cota superior más alta independientemente de su posición en el árbol. Dicho de otra forma, si L es el conjunto de nudos a analizar, escoger aquel $i \in L$, que maximice $\overline{z}^i$.

(b) Escoger aquel nudo que con mayor probabilidad contenga una solución óptima. Aunque esto puede parecer evidente, la razón de hacerlo así estriba en que una vez obtenida una solución óptima, aunque no se puede inmediatamente probar que es el caso, sí se habrá conseguido el máximo valor posible de $\overline{z}_{PE}$. Esto es muy importante para sucesivas podas. Para hacerlo, si $\hat{z}^i \leq \overline{z}^i$, escoger un nudo $i \in L$ que maximice $\hat{z}^i$.

(c) A pesar de que intentar encontrar una solución óptima es muy deseable, a veces puede resultar más práctico tratar de encontrar rápidamente una solución factible $\hat{x}$ tal que $c^T\hat{x} \geq \underline{z}_{PE}$. El criterio

$$\max_{i \in L} \frac{\overline{z}^i - \underline{z}_{PEM}}{\overline{z}^i - \hat{z}^i} \qquad (8.97)$$

que se denomina de **mejora rápida**, trata de conseguir este objetivo. Obsérvese que un nudo i en el que $\hat{z}^i > \underline{z}_{PEM}$ será más adecuado que otro j en el que $\hat{z}^j \leq \underline{z}_{PEM}$. Más aún, los nudos para los cuales la diferencia $\overline{z}^i - \hat{z}^i$ es pequeña deberán tener preferencia. Es de suponer que tales nudos proporcionarán rápidamente una solución factible del problema. El criterio de la mejora rápida se una en bastantes solvers como la opción por defecto a utilizar una vez que se conoce la primera solución factible.

D. Selección de la variable de separación

Una vez ya se ha elegido un nudo *i* a analizar, asociado a este nudo habrá un programa lineal de solución x^i. El siguiente paso, el paso 4 del algoritmo de branch and bound, a llevar a cabo consiste en escoger la variable que va a definir la división que parta de ese nudo i. En lo que sigue restringiremos esas posibilidades de elección al conjunto índice $N^i = \{j \in N: x_j^i \notin Z^1\}$. La evidencia empírica de muchos años de utilización de este procedimiento demuestra la importancia fundamental que tiene el escoger bien un $j \in N^i$. Los criterios para ello suelen variar de unos solvers a otros. Los que exponemos a continuación son los más aceptados y se basan en el cálculo previo a la toma de decisión de unas **degradaciones** o **penalizaciones** en que se puede incurrir en el valor de $\overline{z}^i$ al requerir que una variable x_j que no es entera deba serlo. Veamos pues en que consiste la selección basada en penalizaciones.

Dado el vector solución del programa lineal que está siendo considerado en el nudo i

$$x_{B_i} = \overline{a}_{i0} + \sum_{j=m+1}^{n} \overline{a}_{ij}(-x_j) \quad \text{para} \quad i = 1, \cdots, m \qquad (8.98)$$

y el valor de la función objetivo

$$x_0 = \overline{a}_{00} + \sum_{j=m+1}^{n} \overline{a}_{0j}(-x_j) \qquad (8.99)$$

Obsérvese que en esta última expresión los $\overline{a}_{0j}$, para $j = m+1, \cdots, n$, son los costes reducidos de las variables no básicas. Consideraremos la elección de la variable de separación dentro de dos apartados generales: que esa variable se elige del conjunto de las básicas y del de las no básicas.

(a) **Variables básicas**

Supongamos que en el nudo i alguna variable básica x_p no es entera debiendo serlo, es decir

$$x_p = \bar{a}_{p0} + \sum_{j=m+1}^{n} \bar{a}_{pj}(-x_j) = n_{p0} + f_{p0} \qquad (8.100)$$

donde n_{p0} denota la parte entera de x_p, $\lfloor x_p \rfloor$ y f_{p0} la parte fraccionaria.

Como ya mencionamos en el apartado correspondiente, la división que definirá la variable x_p es

$$\begin{aligned} x_p &\geq n_{p0} + 1 \\ x_p &\leq n_{p0} \end{aligned} \qquad (8.101)$$

Ahora bien, de la teoría de dualidad, y más concretamente del algoritmo dual del simplex, podemos deducir que la imposición de la nueva cota $x_p \geq n_{p0} + 1$ a la variable x_p implicará un empeoramiento del valor de la función objetivo; como la variable básica x_p, realizando una iteración del método dual del simplex, pasaría como mínimo a ser no básica, otra variable no básica, x_j, que estuviese en uno de sus límites, se incrementaría o decrementaría según estuviese en el inferior o superior. Es decir, se reduciría el valor de la función objetivo en una cantidad o penalización que vendría dada por la expresión:

$$P_U = \min_{j,\bar{a}_{pj}<0} (1-f_{p0}) \frac{\bar{a}_{0j}}{-\bar{a}_{pj}} \quad \text{si } x_j = l_j \qquad (8.102)$$

o por

$$P_U = \min_{j,\bar{a}_{pj}>0} (1-f_{p0}) \frac{\bar{a}_{0j}}{-\bar{a}_{pj}} \quad \text{si } x_j = u_j \qquad (8.103)$$

Razonando de manera similar, la imposición de la nueva cota $x_p \leq n_{p0}$ a la variable x_p implicará un empeoramiento del valor de la función objetivo que vendrá dado por la expresión:

$$P_D = \min_{j,\bar{a}_{pj}>0} f_{p0} \frac{\bar{a}_{0j}}{-\bar{a}_{pj}} \quad \text{si } x_j = l_j \qquad (8.104)$$

o por

$$P_U = \min_{j,\bar{a}_{pj}<0} f_{p0} \frac{\bar{a}_{0j}}{-\bar{a}_{pj}} \quad \text{si } x_j = u_j \qquad (8.105)$$

Cualquier solución entera que se pudiese obtener partiendo del nudo *i* estaría por consiguiente acotada superiormente por

$$\max \{x_0 - P_U, x_o - P_D\} \qquad (8.106)$$

Se podrá conseguir una mejor solución desde el nudo *i*, si y sólo, si

$$\min \{P_U, P_D\} < x_0 - z_{PEM} \qquad (8.107)$$

para cada variable básica que no sea entera debiendo serlo. Si esta condición no se cumple se abandona el nudo i pues no tiene interés seguir analizándolo.

Si todas las variables x_p cumplen la condición anterior, las cuestión siguiente a solventar es qué variable de separación se elige. Existen dos principios básicos que cabe considerar para tomar dicha decisión. El primero es elegir aquella con la penalización asociada más pequeña. El segundo, escoger aquella variable con una penalización asociada más grande y comenzar imponiendo el límite opuesto al que determina dicha penalización. Si, por ejemplo, la penalización más grande asociada a una variable x_p es P_U, comenzaremos a analizar el problema que resulte de imponer a esa variable la nueva cota n_{p0} incluyendo en la lista de nudos a analizar más adelante el que resulte de imponer la cota $n_{p0}+1$. La justificación de este proceder es evidente: si se almacenan las peores soluciones, una vez se encuentre una buena, se rechazarán rápidamente buena parte de los nudos que queden en la lista.

Cuando varias de las variables que debiendo ser enteras y no lo son violan la condición anterior, conviene simultáneamente acotar todas esas variables con los límites opuestos a los que violan la condición, almacenando estos últimos en la lista: es probable que sean rechazados muy poco después.

(b) **Variables no básicas**

En el apartado anterior se han analizado las penalizaciones que sobre la función objetivo acarrean la imposición de unos nuevos límites al problema como consecuencia del hecho de que una determinada variable básica, debiendo ser entera, no lo es.

Si una variable x_q es no básica en una determinada solución, ha de ser entera y está en uno de sus límites superior o inferior entero, cualquier cambio en su estado, para seguir siendo entera, deberá serlo en una cantidad como mínimo igual a 1. La penalización por efectuar este cambió está inmediatamente determinada por el coste reducido de esta variable, es decir, $\bar{a}_{0q}$. Si la mejor solución entera factible obtenida hasta ese momento es x_{0c}, la que determina, continua o entera, aquella en la que la variable que nos ocupa toma el valor x_q es x_0 y la penalización $\bar{a}_{0q}$ cumple que $\bar{a}_{0q} \geq x_0 - x_{0c}$, está claro que no interesa moverla del estado en que se encuentra pues no mejoraría x_{0c}: se impondría un nuevo límite inferior o superior a x_q igual al límite en el que estuviese para que en lo sucesivo permaneciese fija. Todo esto, claro está, para cualquier rama de búsqueda de inferior nivel que partiese de algún nudo en el que se hacen estas consideraciones.

La consideración de que las variables no básicas han de ser enteras se puede usar para determinar unas penalizaciones P_U y P_D más estrictas. En efecto, recordemos que al imponer una nueva cota $x_p \geq n_{p0}+1$ a una variable, suponiendo que en una iteración del método dual del simplex esa x_p pasase a ser como mínimo no básica con esa nueva cota, una variable no básica x_q habría de moverse del límite en que estuviese. Si estuviera en su límite inferior o superior, ese incremento vendría dado por la expresión

$$\frac{1-f_{p0}}{-\bar{a}_{pq}} \quad \text{si } \bar{a}_{pq} < 0 \quad \text{y} \quad x_q = l_q \qquad (8.108)$$

o por

$$\frac{1-f_{p0}}{-\bar{a}_{pq}} \quad \text{si } \bar{a}_{pq} > 0 \quad \text{y} \quad x_q = u_q \qquad (8.109)$$

De igual manera, al imponer la nueva cota $x_p \leq n_{p0}$ a la variable, alguna variable no básica x_q variaría su valor en una cantidad dada por la expresión

$$\frac{f_{p0}}{\bar{a}_{pq}} \quad \text{si } \bar{a}_{pq} > 0 \quad \text{y} \quad x_q = l_q \qquad (8.110)$$

o por

$$\frac{f_{p0}}{\bar{a}_{pq}} \quad \text{si } \bar{a}_{pq} < 0 \quad \text{y} \quad x_q = u_q \qquad (8.111)$$

Ahora bien, al considerar las variables x_q no básicas enteras, las cantidades anteriores serán por lo general no enteras. Si son estrictamente mayor que cero y menor que uno, se puede volver a usar el hecho de que x_q debe ser entera, en el sentido de pensar que si cambia su estado, para seguir siendo entera, ese cambio deberá ser una unidad, como mínimo, siendo la penalización de la función objetivo por ese cambio su coste reducido a $\bar{a}_{0q}$. En consecuencia, la penalización por incrementar la variable básica x_q a, al menos $n_{p0}+1$, incrementando o decrementando una variable no básica x_q que debe ser entera, será

$$\max\left\{\bar{a}_{0q}, \bar{a}_{0q}\frac{1-f_{p0}}{-\bar{a}_{pq}}\right\} \quad \text{si } \bar{a}_{pq} < 0 \quad \text{y} \quad x_q = l_q \qquad (8.112)$$

o

$$\max\left\{\bar{a}_{0q}, \bar{a}_{0q}\frac{1-f_{p0}}{-\bar{a}_{pq}}\right\} \quad \text{si } \bar{a}_{pq} > 0 \quad \text{y} \quad x_q = u_q \qquad (8.113)$$

De igual manera, la penalización por decrementar x_p a, al menos, n_{p0}, incrementando o decrementando una variable no básica x_q que debe ser entera, será

$$\max\left\{\bar{a}_{0q}, \bar{a}_{0q}\frac{f_{p0}}{\bar{a}_{pq}}\right\} \quad \text{si } \bar{a}_{pq} > 0 \quad \text{y} \quad x_q = l_q \qquad (8.114)$$

o

$$\max\left\{\bar{a}_{0q}, \bar{a}_{0q}\frac{f_{p0}}{\bar{a}_{pq}}\right\} \quad \text{si } \bar{a}_{pq} < 0 \quad \text{y} \quad x_q = u_q \qquad (8.115)$$

Las nuevas penalizaciones P_U y P_D a utilizar serán pues

$$P_U^* = \min_{j,\bar{a}_{pj}<0}\left\{\begin{array}{ll}\bar{a}_{0j}(1-f_{p0})/(-\bar{a}_{pj}) & \text{si } j\in M\\ \max\left\{\bar{a}_{0j}, \bar{a}_{0j}(1-f_{p0})/(-\bar{a}_{pj})\right\} & \text{si } j\in J\end{array}\right\} \quad \text{si } x_j = l_j \qquad (8.116)$$

o

$$P_U^* = \min_{j,\bar{a}_{pj}>0}\left\{\begin{array}{ll}\bar{a}_{0j}(1-f_{p0})/(-\bar{a}_{pj}) & \text{si } j\in M\\ \max\left\{\bar{a}_{0j}, \bar{a}_{0j}(1-f_{p0})/(-\bar{a}_{pj})\right\} & \text{si } j\in J\end{array}\right\} \quad \text{si } x_j = u_j \qquad (8.117)$$

y

$$P_D^* = \min_{j,\bar{a}_{pj}>0}\left\{\begin{array}{ll}\bar{a}_{0j}f_{p0}/\bar{a}_{pj} & \text{si } j\in M\\ \max\left\{\bar{a}_{0j}, \bar{a}_{0j}f_{p0}/\bar{a}_{pj}\right\} & \text{si } j\in J\end{array}\right\} \quad \text{si } x_j = l_j \qquad (8.118)$$

o

$$P_D^* = \min_{j,\bar{a}_{pj}<0}\left\{\begin{array}{ll}\bar{a}_{0j}f_{p0}/\bar{a}_{pj} & \text{si } j\in M\\ \max\left\{\bar{a}_{0j}, \bar{a}_{0j}f_{p0}/\bar{a}_{pj}\right\} & \text{si } j\in J\end{array}\right\} \quad \text{si } x_j = u_j \qquad (8.119)$$

don J es el conjunto índice de las variables no básicas que han de ser enteras y M el de las no básicas que no han de ser enteras. Estas penalizaciones más estrictas permiten encontrar una buena solución entera sin necesidad de analizar todos los nudos. Igual que antes, cualquier solución entera que se pudiese obtener a partir del nudo i estaría acotada superiormente por

$$\max\{x_0 - P_U^*, x_0 - P_D^*\} \qquad (8.120)$$

El nudo i se podrá desechar si, para cualquier x_p,

$$\min\{P_U^*, P_D^*\} \geq x_0 - z_{PEM} \qquad (8.121)$$

El resumen de todas estas consideraciones es que, incorporando los sencillos cálculos descritos al analizar un nudo, es posible determinar unas penalizaciones que permiten desechar muchas alternativas de separación: esto permite reducir substancialmente el tiempo de cálculo de obtención de la solución óptima de un programa entero mixto por el procedimiento de branch and bound.

8.8 Solvers de Programación Entera Mixta: MOMIP [Fuente95]

Los solvers de programación entera mixta son programas informáticos capaces de resolver problemas de optimización del tipo

$$\begin{aligned} \max \quad & (c^T x + h^T y) \\ \text{s.t.} \quad & Ax + Gy \le b \\ & l_x \le x \le u_x \\ & l_y \le y \le u_y \\ & x \in R^n \\ & y \in Z^p \end{aligned} \qquad (8.122)$$

Existen multitud de solvers que implementan los algoritmos, básicamente de tipo branch and bound, para la resolución de este tipo de problemas: MOMIP, LINDO, MINTO, ...

A continuación pasaremos a analizar la estructura general de un solver de programación entera mixta, como se le debe proporcionar la especificación del problema y finalmente analizaremos el solver de programación entera mixta utilizado en esta tesis, MOMIP.

8.8.1 Ficheros de Especificación de Problemas

La especificación del problema a resolver se realiza en la mayoría de solvers de programación entera mixta y general en los solvers de programación lineal y programación entera mediante un fichero con formato MPS ("Mathematical Programming Systems"). Este formato fue creado por IBM. Un fichero con formato MPS contiene por lo tanto la especificación del problema a resolver por el solver. El fichero MPS se divide en una serie de secciones delimitada por las siguiente palabras clave:

NAME
ROWS
...
INT
...
RHS
...
RANGES
...
BOUNDS
...
ENDATA

El formato MPS es sin duda el más extendido en el mundo entre los códigos comerciales que resuelven problemas de optimización. El detalle del formato de este fichero se puede encontrar en el manual de usuario de cualquier solver de optimización que lo utilice o en cualquier libro de optimización que trate la optimización lineal y entera como por ejemplo el libro de De la Fuente O'Connor [O'Connor95].

8.8.2 Estructura General de un Solver de Programación Entera Mixta

La estructura de un solver de programación entera mixta es la siguiente:

1ª Etapa: "*Entrada de datos*"

◊ Lectura de las especificaciones del proceso de optimización.

◊ Lectura del fichero de datos del problema de optimización a resolver en formato MPS.

2ª Etapa: *"Resolución inicial del problema de programación entero mixto"*

◊ Resolución del problema mediante relajación lineal utilizando el método simplex revisado.
◊ Presentación de los resultados
◊ Si el problema tiene variables enteras, se procede a la siguiente etapa. En caso contrario, finaliza el programa.

3ª Etapa: *"Aplicación del algoritmo branch and bound aplicando relajaciones lineales"*

◊ Comprobación de la solución óptima del programa lineal, rechazándose el nudo y por tanto no generándose a partir de él ninguna rama si se cumple alguna de las siguientes condiciones:

(a) La solución del problema lineal no es factible
(b) El valor de la función objetivo del problema lineal es peor que la mejor solución entera obtenida hasta el momento
(c) La solución del problema lineal es entera.

◊ Cálculo de las penalizaciones para cada variable básica no entera que debería serlo. También se comprueba la existencia de separaciones forzadas para variables no básicas. Como variable de separación se escoge aquella con la mayor penalización. La siguiente acción se lleva a cabo en la dirección opuesta a la máxima penalización. La variable de máxima penalización se añade a la lista de las que se analizarán más adelante, generándose dos nuevas ramas del árbol numerativo.
◊ Resolución de la relajación lineal del nudo del árbol numerativo siguiendo la regla LIFO. Para ello se utiliza el método primal del simplex. Una vez resuelto el problema se pueden dar alguna de las siguientes situaciones:

(d) Si la solución conseguida es entera factible, se escoge otro nudo de la lista y se vuelve a comenzar el ciclo primal.
(e) En caso contrario, se vuelven a calcular las penalizaciones de cada variable básica.

4ª Etapa: *"Presentación de los resultados del problema de optimización"*

◊ Presentación del valor de la función objetivo
◊ Presentación del valor de las condiciones
◊ Presentación del valor de las variables

8.8.3 El Optimizador MOMIP

MOMIP es un solver de optimización desarrollado en C++, que permite resolver problemas de optimización entera mixta de tamaño medio utilizando el algoritmo de branch and bound. Este software ha sido desarrollo en un instituto de investigación austríaco denominado IIASA ("International Institute for Applied Systems Analysis"). [Ogryczak96]

El algoritmo de branch and bound implementado por MOMIP es el siguiente:

Algoritmo de Branch and Bound de MOMIP

Paso 1. Definimos el nodo 0 mediante la relajación lineal del problema original y la solución óptima disponible de dicho problema. Si todas las variables de la solución son enteras, la búsqueda ha finalizado. Fijamos el contador de nodos examinados a cero, n = 0, fijamos el valor de corte inicial, elegimos el nodo 0 como el nodo de bifurcación, k = 0, y escogemos la variable de bifurcación.

Paso 2. Definimos los nodos n+1 y n+2 como los subproblemas del nodo k de acuerdo con la variable de bifurcación preseleccionada, n = n + 2.

Paso 3. Optimizamos el problema asociado con el nodo n+1. Si el problema no es factible, entonces lo eliminamos de posteriores análisis. Mientras que si la solución satisface las condiciones de optimalidad, entonces almacenamos la mejor solución entera encontrada hasta ahora, modificamos el valor de corte y eliminamos todos aquellos nodos que podamos de la lista. Finalmente, si la solución no satisface las condiciones de optimalidad, entonces seleccionamos una nueva posible variable de bifurcación.

Paso 4. Optimizamos el problema asociado con el nodo n+2. Si el problema no es factible, entonces lo eliminamos de posteriores análisis. Mientras que si la solución satisface las condiciones de optimalidad, entonces almacenamos la mejor solución entera encontrada hasta ahora, modificamos el valor de corte y eliminamos todos aquellos nodos que podamos de la lista. Finalmente, si la solución no satisface las condiciones de optimalidad, entonces seleccionamos una nueva posible variable de bifurcación.

Paso 5. Si la lista de nodos a analizar está vacía, la búsqueda ha finalizado. La mejor solución entera es la óptima. Si no existe ninguna solución entera, el problema no tiene solución entera. En caso de que la lista no esté vacía, escogemos el siguiente nodo de bifurcación k de la lista y eliminados el nodo actual de la misma, y volvemos al paso 2.

El valor de corte inicial se define en MOMIP por defecto a ∞ para los problemas de maximización y a $-\infty$ para los problemas de minimización, aunque el usuario puede especificar un valor diferente mediante el parámetro CUTOFF. La búsqueda se restringe entonces a soluciones enteras que hagan que el valor de la función objetivo sea mejor que el parámetro CUTOFF. Cuando se encuentra una solución entera, entonces el parámetro CUTOFF se fija a:

$$CUTOFF = V - MINMAX \times OPTEPS \times |V|$$

donde: V es el valor de la función objetivo asociado a la solución entera, OPTEPS es la tolerancia relativa asociada al valor óptimo y MINMAX es 1 para la minimización y -1 para la maximización.

8.9 Aplicación de la Programación Separable a la Generación de Envolventes

- **Planteamiento del problema**

Para poder calcular las envolventes en tiempo real pero sin aumentar la incertidumbre total lo que hacemos es truncar la expresión exacta presentada en el apartado anterior. Este truncamiento nos permite calcular las envolventes del sistema en el instante k tomando una ventana de longitud L que se va desplazando a medida que pasa el tiempo:

$$x_n = A^L x_{n-L} + A^{L-1} B u_{n-L} + \cdots + B u_{n-1}$$

de manera que se calcula el valor de las envolventes en el paso k a partir de la incertidumbre en los parámetros:

$$A \in \left[A^-, A^+\right] \quad y \quad B \in \left[B^-, B^+\right]$$

y a partir de la incertidumbre en los estados al inicio de la ventana:

$$x_{k-L} \in \left[x^-_{k-L}, x^+_{k-L}\right]$$

con lo cual el problema del cálculo de las envolventes para cada instante de tiempo k consiste en la resolución de los dos problemas de optimización siguientes:

$x_n^+ = \max\left(A^L x_{n-L} + A^{L-1} B u_{n-L} + \cdots + B u_{n-1}\right)$ $s.t.: x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$	$x_n^- = \min\left(A^L x_{n-L} + A^{L-1} B u_{n-L} + \cdots + B u_{n-1}\right)$ $s.t.: x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$

Se puede demostrar analíticamente y mediante simulaciones que para una longitud L de la ventana del orden del tiempo de establecimiento del sistema no existe diferencia entre las envolventes calculadas por el método exacto y el método aproximado.

La idea de utilizar una ventana soluciona por lo tanto el problema del crecimiento del cálculo a realizar a medida que aumenta el tiempo a la vez que garantiza que el resultado obtenido se aproxima al resultado exacto. El problema que queda abierto y que se analiza en el siguiente apartado es como se resuelve el problema de optimización planteado.

- **Resolución del problema de optimización utilizando programación entera mixta**

Centrémonos en un uno de los dos problemas de optimización que debemos resolver: la determinación del valor de la envolvente máxima en un determinado instante de tiempo:

$$x_n^+ = \max\left(A^L x_{n-L} + A^{L-1} B u_{n-L} + \cdots + B u_{n-1}\right)$$
$$s.t.: x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$$
$$A \in \left[A^-, A^+\right]$$
$$B \in \left[B^-, B^+\right]$$

Para simplificar la explicación, y sin pérdida de generalidad, situemos en un caso concreto:
- un sistema de primer orden: A = a y B = b
- una ventana temporal: L = 2

con lo cual el problema de optimización a resolver se transforma en:

$$x_n^+ = \max\left(a^2 x_{n-2} + abu_{n-2} + bu_{n-1}\right)$$
$$s.t.: x_{n-2} \in \left[x_{n-2}^-, x_{n-2}^+\right]$$
$$a \in \left[a^-, a^+\right]$$
$$b \in \left[b^-, b^+\right]$$

Teniendo en cuenta que para un determinado instante de tiempo u_{n-2} y u_{n-1} serán valores conocidos:

$$c_1 = u_{n-2}$$
$$c_2 = u_{n-1}$$

y realizando el siguiente cambio de variables:

$$x_1 = x_{n-2}$$
$$x_2 = a$$
$$x_3 = b$$

El problema de optimización se puede reescribir como, el problema de maximizar la función de tres variables $y=f(x_1,x_2,x_3)$ dentro del cubo determinado por las cotas de dichas variables:

$$x_n^+ = \max\left(x_2^2 x_1 + c_1 x_2 x_3 + c_2 x_3\right)$$
$$s.t.: x_1 \in \left[x_1^-, x_1^+\right] = [x^-, x^+]$$
$$x_2 \in \left[x_2^-, x_2^+\right] = [a^-, a^+]$$
$$x_3 \in \left[x_3^-, x_3^+\right] = [b^-, b^+]$$

En general, este problema será un problema de optimización no lineal y no convexo. Este tipo de problema presenta el problema de los **óptimos locales**. La mayoría de algoritmos para este tipo de problemas no pueden más que garantizar un óptimo local, puesto que se basan en una semilla como punto de arranque. Si dicha semilla se encuentra cerca de un óptimo local el algoritmo convergerá hacia el mismo.
Tal como hemos visto en el capítulo anterior para el caso de tener un **problema separable** y en el que aparecen **productos entre variables** elevadas a productos se puede aplicar la programación entera mixta como un **método de resolución aproximado**, pero que garantiza que se alcanza el óptimo global.

- **Transformación de la función objetivo en una función lineal mediante un cambio de variable**

Para poder aplicar el método de la programación entera mixta en primer lugar se debe aplicar un cambio de variable que transforme la función objetivo en una función lineal. Si introducimos las variables z_1 y z_2 de la siguiente manera:

$$z_1 = x_2^2 x_1$$
$$z_2 = x_2 x_3$$

La función objetivo se transforma en una función lineal:

$$z_1+c_1z_3+c_2x_3$$

quedando el problema de optimización transformado en:

$$x_n^+ = \max \quad (z_1 + c_1 z_2 + c_2 x_3)$$
$$s.t.:$$
$$z_1 = x_2^2 x_1$$
$$z_2 = x_2 x_3$$
$$x_1 \in \left[x_1^-, x_1^+\right]$$
$$x_2 \in \left[x_2^-, x_2^+\right]$$
$$x_3 \in \left[x_3^-, x_3^+\right]$$
$$z_1 \in \left[z_1^-, z_1^+\right]$$
$$z_2 \in \left[z_2^-, z_2^+\right]$$

Se observa que el nuevo problema de optimización obtenido presenta una función objetivo lineal pero unas restricciones no lineales.

- **Linealización de las restricciones formadas por los productos**

Las restricciones no lineales que han aparecido están formadas por los diferentes productos entre variables que contenía la función objetivo. Estos productos se pueden linealizar aplicando un cambio de variables logarítmico y una linealización a trozos. En primer lugar, vamos a tomar logaritmos de cada uno de los productos a linealizar:

$$\log z_1 = 2\log x_2 + \log x_1$$
$$\log z_2 = \log x_2 + \log x_3$$

Si ahora realizamos los siguientes cambios de variable:

$$w_1 = \log z_1$$
$$w_2 = \log z_2$$
$$v_1 = \log x_1$$
$$v_2 = \log x_2$$
$$v_3 = \log x_3$$

las anteriores expresiones logarítmicas se transformarán en:

$$w_1 = v_1 + 2v_2$$
$$w_2 = v_2 + v_3$$

De forma el problema de optimización quedará transformado en:

$$x_n^+ = \max \quad (z_1 + c_1 z_2 + c_2 x_3)$$
s.t.:
$$w_1 = v_1 + 2v_2$$
$$w_2 = v_2 + v_3$$
$$w_1 = \log z_1$$
$$w_2 = \log z_2$$
$$v_1 = \log x_1$$
$$v_2 = \log x_2$$
$$v_3 = \log x_3$$
$$x_1 \in \left[x_1^-, x_1^+\right]$$
$$x_2 \in \left[x_2^-, x_2^+\right]$$
$$x_3 \in \left[x_3^-, x_3^+\right]$$
$$z_1 \in \left[z_1^-, z_1^+\right]$$
$$z_2 \in \left[z_2^-, z_2^+\right]$$
$$v_1 \in \left[v_1^-, v_1^+\right] = \left[\log x_1^-, \log x_1^+\right]$$
$$v_2 \in \left[v_2^-, v_2^+\right] = \left[\log x_2^-, \log x_2^+\right]$$
$$v_3 \in \left[v_3^-, v_3^+\right] = \left[\log x_3^-, \log x_3^+\right]$$
$$w_1 \in \left[w_1^-, w_1^+\right] = \left[\log x_1^- (x_2^-)^2, \log x_1^+ (x_2^+)^2\right]$$
$$w_2 \in \left[w_2^-, w_2^+\right] = \left[\log x_2^- x_3^-, \log x_2^+ x_3^+\right]$$

Finalmente, lo que queda es linealizar los logaritmos a trozos utilizando una linealización tipo λ o tipo δ, tal como se explicado al principio de este capítulo.

Utilizando una linealización a trozos tipo λ con una rejilla de seis puntos para cada uno de los logaritmos, por ejemplo, la restricción $w_1 = \log z_1$
se transformará en:

$$w_1 = w_{1a}\lambda_{1a} + w_{1b}\lambda_{1b} + w_{1c}\lambda_{1c} + w_{1d}\lambda_{1d} + w_{1e}\lambda_{1e} + w_{1f}\lambda_{1f}$$
$$z_1 = z_{1a}\lambda_{1a} + z_{1b}\lambda_{1b} + z_{1c}\lambda_{1c} + z_{1d}\lambda_{1d} + z_{1e}\lambda_{1e} + z_{1f}\lambda_{1f}$$
$$\lambda_{1a} - I_{a1} < 0$$
$$\lambda_{1b} - I_{a1} - I_{b1} < 0$$
$$\lambda_{1c} - I_{b1} - I_{c1} < 0$$
$$\lambda_{1d} - I_{c1} - I_{d1} < 0$$
$$\lambda_{1e} - I_{d1} - I_{e1} < 0$$
$$\lambda_{1f} - I_{e1} < 0$$
$$\lambda_{1a} + \lambda_{1b} + \lambda_{1c} + \lambda_{1d} + \lambda_{1e} + \lambda_{1f} = 1$$
$$I_{a1} \in \{0,1\}$$
$$I_{b1} \in \{0,1\}$$
$$I_{c1} \in \{0,1\}$$
$$I_{d1} \in \{0,1\}$$
$$I_{e1} \in \{0,1\}$$

donde:

$\{w_{1a}, w_{1b}, w_{1c}, w_{1d}, w_{1e}, w_{1f}\} \in [w_1^-, w_1^+]$ y $\{z_{1a}, z_{1b}, z_{1c}, z_{1d}, z_{1e}, z_{1f}\} \in [z_1^-, z_1^+]$ son las rejillas de punto correspondientes a la variables w_1 y z_1 relacionadas a través de logaritmo,
$\lambda_{1a}, \lambda_{1b}, \lambda_{1c}, \lambda_{1d}, \lambda_{1e}, \lambda_{1f}$ son variables reales de interpolación
$I_{a1}, I_{b1}, I_{c1}, I_{d1}, I_{e1}$ son variables enteras binarias que garantizan las condiciones de adyacencia

Análogamente se procedería a la linealización del resto de logaritmos. Una vez linealizados todos los logaritmos se habría traducido el problema original no lineal y no convexo en un problema lineal a trozos que para garantizar que se determine el óptimo global debería de ser resuelto utilizando un solver de programación entera mixta.

- **Resolución del problema mediante un solver de programación entera mixta**

Una vez linealizada la función objetivo y las restricciones mediante aproximaciones lineales a trozos, el problema obtenido es un problema de optimización entero mixto. Este problema se puede resolver mediante algoritmos de Branch and Bound combinadas con las técnicas de resolución de problemas de optimización lineales del tipo simplex. Todos los solvers de programación entera mixta utilizan como formato de entrada del problema el formato de fichero MPS. Por lo tanto, una vez obtenido el problema de optimización lineal a trozos se debería de traducir al formato MPS, para que pudiera ser resuelto por un solver de programación entera mixta.

8.10 Implementación de un Generador de Envolventes basado en esta técnica

Para poder resolver el problema de la generación de envolventes utilizando programación entera mixta se debe desarrollar un programa informático que realice las siguientes funciones:

- obtención de la función objetivo como suma de productos de variables elevadas a potencias
- linealización de la función objetivo mediante cambios de variable
- linealización de las restricciones mediante la aplicación de cambios de variable logarítmicos y linealización a trozos
- introducción de variables binarias y planteamiento del problema como un problema de optimización entero mixto
- traducción del problema al formato MPS para poder ser resuelto por alguno de los solvers estándar de programación entera mixta
- solución del problema mediante el solver de programación entera mixta
- obtención de los resultados

Veamos como se podrían conseguir cada una de las funcionalidades anteriormente mencionadas.

8.10.1 Estructura del Generador de Envolventes

El sistema informático que se ha desarrollado para generar las envolventes utilizando la técnica de optimización global descrita en este capítulo está compuesto de los siguientes bloques funcionales:

- manipulador algebraico: MAPLE
- linealizador: programa desarrollado en C
- traductor de formato LINDO a formato MPS
- solver del problema de optimización entera mixta: MOMIP
- extractor de la solución: programa desarrollado en C
- manipulador del fichero MPS: programa desarrollado en C
- sistema de generación de envolventes en tiempo real: programa desarrollado en C

La interrelación entre estos bloques funcionales se muestra en la Fig. 8.6.

8.10.2 Descripción del funcionamiento de los módulos funcionales

Una vez descrita la estructura del sistema informático que se ha desarrollado para la generación de envolventes en tiempo real, vamos a describir cada una de las unidades funcionales que se han identificado.

- **El manipulador algebraico**

El primer problema que debe resolver nuestro sistema informático es la obtención de una expresión para la función objetivo manipulada en forma de sumas de productos de variables elevadas a potencias. Para conseguir realizar este proceso de forma automática se precisa de un manipulador algebraico. Se podría haber construido dicho manipulador, pero en el mercado existen manipuladores algebraicos muy potentes con lo cual resulta innecesario su desarrollo. El manipulador algebraico que se ha utilizado es el programa matemático MAPLE.

- **El linealizador**

Con la expresión de la función objetivo manipulada, el siguiente módulo funcional, que lo hemos denominado linealizador, extraerá los productos de la función objetivo mediante un cambio de variable y los linealizará mediante un cambio de variable logarítmico previo y una linealización a trozos utilizando

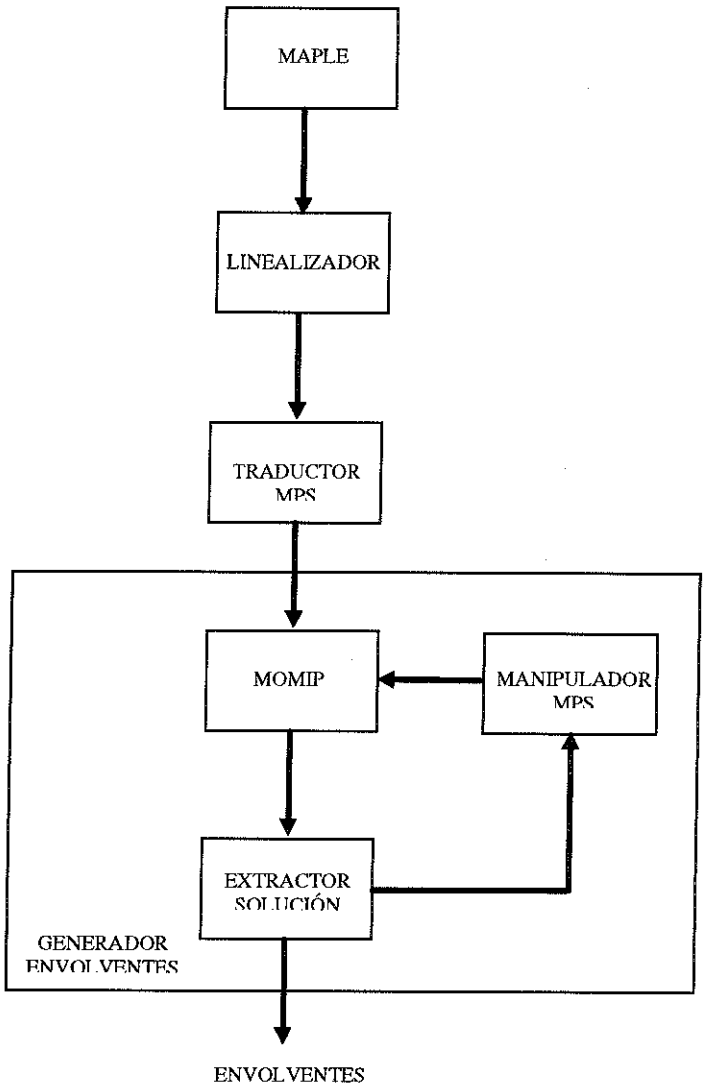


Fig. 8.6 Generación de las envolventes utilizando programación entera mixta

variables binarias. A partir de del problema linealizado construirá un fichero tipo LINDO que contendrá dicho problema. Este módulo funcional ha sido realizado mediante un programa en C.

El funcionamiento del mismo es el siguiente:

1. A partir del fichero MAPLE que contiene la expresión de la función objetivo linealizada, el linealizador va extrayendo cada uno de los productos y los va almacenando en una Tabla de Productos.
2. A partir de la Tabla de Productos se aplica sobre cada uno de ellos la linealización a trozos previo cambio logarítmico que nos separa cada uno de las variables del producto.
3. A continuación se introducen las variables binarias y se calculan las cotas superior e inferior de cada una de las variables del problema
4. Finalmente se construye un fichero tipo LINDO que contiene el problema de optimización traducido a un problema de programación entera mixta apto para ser resuelto con el solver adecuado.

- **El traductor de formato LINDO a formato MPS**

La mayoría de solvers de programación entera mixta que existen en el mercado utilizan como formato de entrada de los problemas a resolver el formato MPS. El solver que se ha utilizado en este proyecto denominado MOMIP utiliza también dicho formato. Por lo tanto, se ha tenido que añadir al sistema informático de generación de envolventes mediante programación entera mixta un traductor que tradujera del formato LINDO obtenido por MAPLE al formato MPS. Este módulo ha sido desarrollado también en C.

El funcionamiento del traductor es el siguiente:

1. En primer lugar, el traductor lee completamente el fichero que contiene el problema de optimización entera mixta contenido en el fichero LINDO almacenando el contenido de dicho fichero en diferentes tablas: Tabla de Coeficientes, Tabla de Variables, Tabla de Coeficientes RHS, Tabla de Variables Enteras y Tabla de Cotas Mínimas/Máximas de las Variables.
2. A partir del contenido de las tablas anteriores se construye el fichero MPS que contenga el problema.

- **El traductor de formato MPS a formato LINDO**

El formato MPS es adecuado para los solvers de programación entera mixta pero no para el usuario. Para poder leer y ver de forma inteligible el contenido de un fichero MPS se ha desarrollado otro traductor que permite obtener un problema entero mixto especificado mediante un fichero MPS en el formato LINDO. Este módulo ha sido desarrollado en C.
El funcionamiento del traductor es el siguiente:

1. En primer lugar, el traductor lee completamente el fichero que contiene el problema de optimización entera mixta contenido en el fichero MPS almacenando el contenido de dicho fichero en diferentes tablas: Tabla de Coeficientes, Tabla de Variables, Tabla de Coeficientes RHS, Tabla de Variables Enteras y Tabla de Cotas Mínimas/Máximas de las Variables.
2. A partir del contenido de las tablas anteriores se construye el fichero LINDO que contenga el problema.

- **El solver de problemas enteros mixtos**

Dicho módulo funcional tiene como finalidad encontrar la solución a un problema de programación entera mixta. En el mercado existen multitud de solvers que permiten resolver este tipo de problemas es por ello que no se ha desarrollado en este proyecto un solver específico sino que se ha utilizado uno de los existentes. El solver que se ha utilizado se denomina: MOMIP (*"Modular Optimizer for Mixed Integer Programming"*).

El funcionamiento de este programa es el siguiente:

1. Se ejecuta el programa indicándole cual es el fichero en formato MPS que contiene el problema de programación entera mixta a resolver.
2. Después de procesar dicho fichero, resuelve el problema contenido en el mismo devolviendo un fichero que contiene la solución al problema. Se trata de un fichero propio del programa MOMIP.

- **El extractor de soluciones de fichero solución de MOMIP**

Una vez se ha resuelto el problema de optimización entera mixta con el programa MOMIP devolviendo la solución del mismo en un fichero solución, se precisa de un módulo funcional que capture la solución contenida en dicho fichero para su utilización en el cálculo de las envolventes.

El funcionamiento de este módulo es el siguiente:

1. Se analiza el fichero solución hasta extraer la solución al problema de programación entera mixta.
2. Una vez localizada y extraída dicha solución se envía al módulo de cálculo de envolventes.

- **El modificador de ficheros MPS**

Resuelto el problema de optimización para el instante actual para cada una de las variables de estado, se debe preparar cada uno de los ficheros que contienen los problemas de optimización para que contengan el problema que nos proporcionará el valor de las envolventes en el siguiente instante de tiempo. Esta es la misión del módulo modificador de ficheros MPS

El funcionamiento de este módulo es el siguiente:

1. A partir de un fichero MPS que contiene el problema de optimización que nos proporciona el valor de la envolvente máxima/mínima para una de las variables de estado en el instante actual, se debe modificar dichero fichero para que contenga el problema de optimización que nos proporcionará el valor de las envolventes máxima/mínima en el instante siguiente.
2. Las modificaciones que se deben introducir al problema son las siguientes: modificar las constantes de la función objetivo reflejando el valor actual de las entradas al proceso y modificar las cotas de las variables de estado de forma que contengan el valor de las mismas L pasos hacia atrás. Al cambiar el valor de las variables de estado también cambiarán el valor de las cotas de los productos donde intervengan dichas variables.

- **El generador de envolventes**

Este es el módulo que utiliza el resto de módulos para generar para cada instante de tiempo el valor de la envolvente máxima y mínima mediante la resolución de los correspondientes problemas de programación entera mixta

8.10.3 Aplicación sobre varios ejemplos

Para comprobar el correcto funcionamiento del generador de envolventes basado en programación entera mixta se han utilizados ejemplos que ya han sido utilizados en el capítulo 3 de esta tesis: un ejemplo basado en un sistema de segundo orden y un ejemplo real basado en el servosistema de posicionamiento de los alabes de una turbina de gas utilizada en el proyecto ESPRIT TIGER.

A. Ejemplo de aplicación a un sistema de segundo orden

El sistema de segundo orden que hemos escogido tiene el siguiente modelo en el espacio de estados discreto

$$x_{k+1} = Ax_k + Bu_k$$

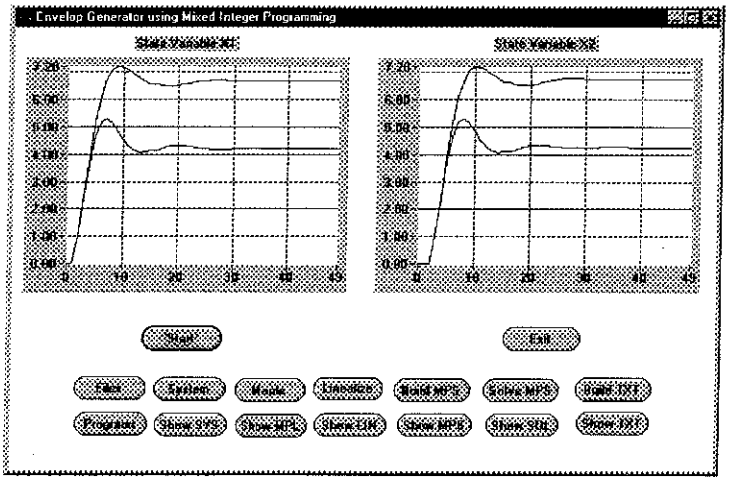
con los siguientes valores nominales para sus parámetros:

$$A_o = \begin{pmatrix} 0 & -0.9048 \\ 1 & 1.8953 \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

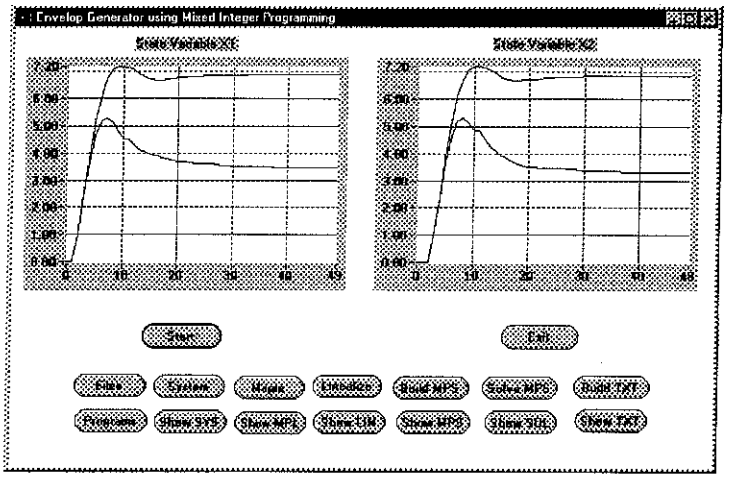
La única matriz que tiene parámetros acotados en intervalos es A, siendo su valores máximos y mínimos, respectivamente:

$$A_{max} = \begin{pmatrix} 0 & -0.9038 \\ 1 & 1.8963 \end{pmatrix} \qquad \text{y} \qquad A_{min} = \begin{pmatrix} 0 & -0.9058 \\ 1 & 1.8943 \end{pmatrix}$$

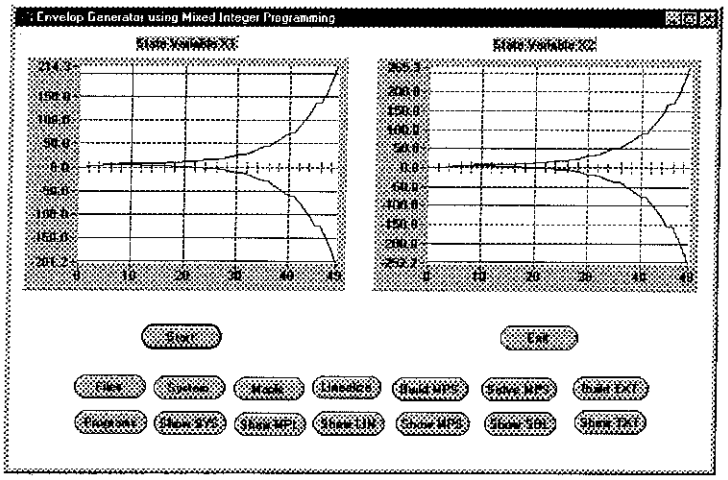
Utilizando el generador de envolventes construido utilizando la técnica de la programación entera mixta las envolventes generadas para este sistema para diferentes longitudes de ventana se muestran en la Fig. 8.7.



(a) Longitud de Ventana L =15



(b)Longitud de Ventana L =10



(c) Longitud de Ventana L =5

Fig. 8.7 Generación de las envolventes
del sistema de segundo orden
para diferentes longitudes de ventana

B. Un ejemplo real: el sistema TIGER

El modelo del servosistema de posicionamiento de los alabes se puede reescribir utilizando la representación normalizada en el espacio de estado como:

$$\mathbf{x}(k+1) = A\cdot\mathbf{x}(k) + B\cdot\mathbf{u}(k)$$

donde:

$$A = \begin{pmatrix} a_1 - a_2 s & -a_2 - s_3 \\ s_1 & 1 \end{pmatrix} = \begin{pmatrix} c_1 & -0.25 \\ c_2 & 1 \end{pmatrix}$$

y

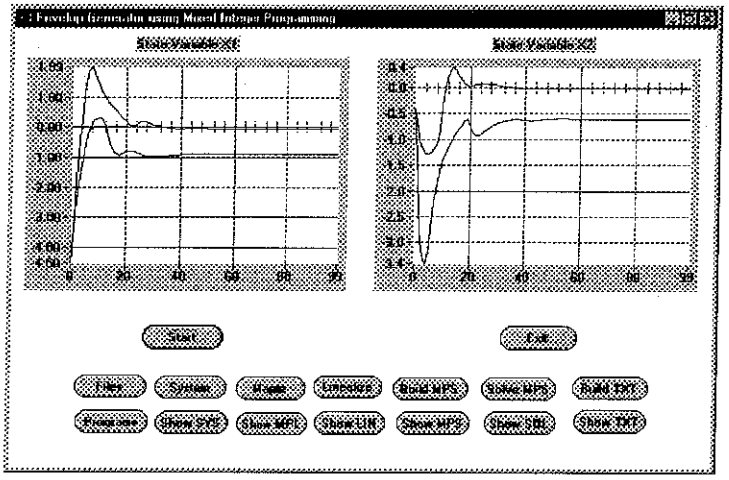
$$B = \begin{pmatrix} 1 & -a_2 s_2 \\ 0 & s_2 \end{pmatrix} = \begin{pmatrix} 1 & c_3 \\ 0 & c_4 \end{pmatrix}$$

siendo el valor de los parámetros inciertos:

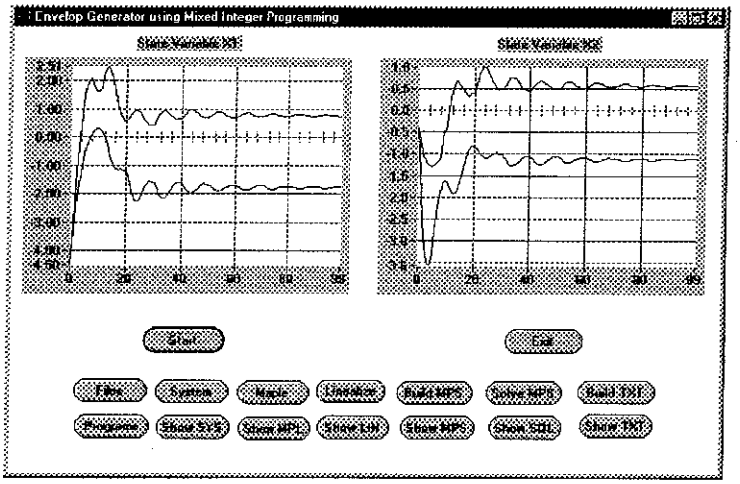
$c_1 \in [0.6797,\ 0.7335]$ $\quad$ $c_2 \in [0.1032, 0.3372]$

$c_3 \in [-0.0212, -0.0035]$ $\quad$ $c_4 \in [0.0155,\ 0.09232]$

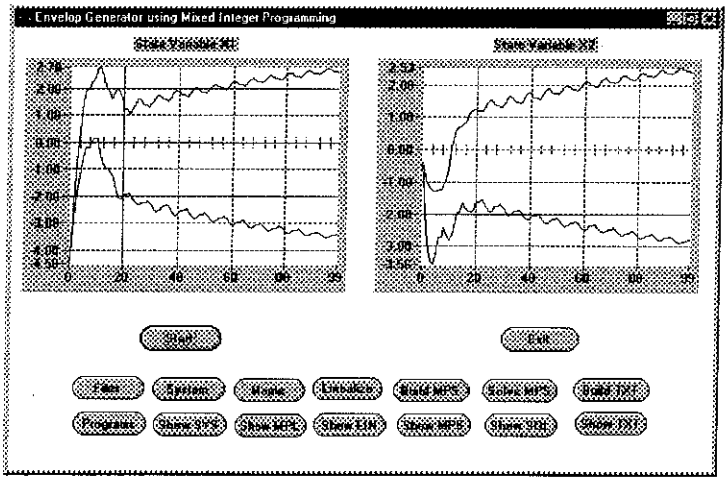
Utilizando el generador de envolventes construido utilizando la técnica de la programación entera mixta las envolventes generadas para este sistema para diferentes longitudes de ventana se muestran en la Fig. 8.8.



(a) Longitud de Ventana L =25



(b) Longitud de Ventana L =10



(c) Longitud de Ventana L =7

Fig. 8.8 Generación de las envolventes
del sistema TIGER para diferentes longitudes de ventana

8.11 Conclusiones

En este capítulo se ha propuesto resolver el problema de programación global asociado al problema de generación de envolventes mediante el nuevo algoritmo presentado en esta tesis, utilizando la propiedad de separabilidad de la función objetivo. La programación separable es un clásico dentro del campo de la optimización, pudiéndose encontrar en la mayoría de los libros de optimización.

El método de optimización basado en la programación separable resulta atractivo puesto que utiliza como método de optimización final el método simplex, que es uno de los métodos más conocidos y utilizados, combinándolo con una búsqueda binaria de tipo branch and bound, pudiéndose combinar generalizando a un problema de programación entera mixta.

Con esta técnica se han obtenido buenos resultados al aplicarla a la generación de envolventes, pero la dificultad aparece en todo el tratamiento simbólico asociado con la transformación del problema original separable pero no lineal, en un problema lineal a trozos formulado como un problema de programación entera mixta y bajo el formato adecuado para que el optimizador que implementa dicho algoritmo pueda resolverlo correctamente. Así mismo, una vez obtenido el óptimo, debe de reformularse todo el problema, linealización incluida, para la determinación el siguiente óptimo asociado al siguiente instante de tiempo. Todo ello hace que el método sea excesivamente lento y complejo para poder ser aplicado a la generación de envolventes en tiempo real, aunque los resultados obtenidos son buenos, pudiéndose determinar el óptimo con la precisión deseada sin más que decidir cuantos puntos se utilizarán en la rejilla de linealización.

Comparando está técnica con la presentada en el capítulo anterior, ambas técnicas como, se ha enfatizado en los títulos correspondientes a ambos capítulos no son técnicas de optimización global generales, sino que son técnicas que aprovechan la estructura particular del problema de optimización que aparece al aplicar el nuevo algoritmo de optimización de envolventes a un sistema lineal invariante con el tiempo, pero con parámetros inciertos cuyo valor esta acotado dentro de un intervalo.

Otro punto de contacto entre ambos algoritmos y en general entre la mayoría de algoritmos de optimización global, tal como hemos comentado, en el capítulo donde se presentaban las características principales de los problemas de optimización global, es la utilización de la búsqueda mediante el algoritmo de branch and bound.

Capítulo 9

Capítulo 9:

OPTIMIZACIÓN GLOBAL MEDIANTE BRANCH AND BOUND Y ALGEBRA DE INTERVALOS

En este capítulo se presenta el álgebra de intervalos aplicada a la optimización global de funciones utilizando el algoritmo branch and bound. Los resultados que se presentan básicamente proceden de los trabajos de Ratscheck [Ratscheck84] Ratscheck[88], Hansen [Hansen92] y Kearfott [Kearfott96], matemáticos que han desarrollado un conjunto de algoritmos basados en el álgebra de intervalos clásica o de Moore [Moore66] junto con el algoritmo de branch and bound para determinar el óptimo global de una función.

A diferencia de las técnicas de optimización global presentadas en los capítulos anteriores que explotaban la estructura particular del problema, las técnicas de optimización global basadas en el álgebra de intervalos y en el algoritmo de branch and bound son técnicas de aplicación más general que son aplicables a una gama muy amplia de problemas de optimización, tanto con restricciones como sin restricciones.

9.1 Introducción al Álgebra de Intervalos Clásica

El álgebra de intervalos clásica fue introducida por R.E. Moore [Moore66], por ello también se la conoce como álgebra de Moore. Dicha álgebra en la definición de una aritmética dentro del conjunto de los intervalos cerrados reales. Si $X=[x^-,x^+]$ y $Y=[y^-,y^+]$, entonces las cuatro operaciones básicas de dicha aritmética se definen mediante

$$X \circ Y = \{x \circ y | x \in X, y \in Y\} \quad \text{para} \quad \circ \in \{+,-,*,/\} \tag{9.1}$$

de donde, el resultado de la aplicación de cada una de las operaciones básicas sobre los intervalos X e Y proporciona otro intervalo Z, que se corresponde con el rango de valores que puede tomar la operación de cada punto del intervalo X con cada punto del intervalo Y.
Aunque la definición anterior caracteriza dichas operaciones matemáticamente, la utilidad del álgebra de intervalos se debe a las definiciones operacionales de dichas operaciones sobre intervalos

$$X+Y=[x^-+y^-,x^++y^+]$$

$$X-Y=[x^--y^+,x^+-y^-]$$

$$X*Y=[\min\{x^-y^-,x^-y^+,x^+y^-,x^+y^+\},\max\{x^-y^-,x^-y^+,x^+y^-,x^+y^+\}]$$

$$\frac{1}{X}=[\frac{1}{x^+},\frac{1}{x^-}] \quad \text{si} \quad x^->0 \quad \text{o} \quad \text{si} \quad x^+<0$$

$$X/Y=X*\frac{1}{Y}$$

Es obvio, a partir de la definiciones anteriores que ni el recíproco de un intervalo ni la división de intervalos están definidas cuando $0 \in Y$. Sin embargo, bajo determinadas circunstancias puede ser útil poder trabajar con cocientes de intervalos que contienen el cero, para ello existe una álgebra de intervalos extendida o de Kahan-Novoa-Ratz.

9.2 Extensión Intervalar de Funciones

Una aplicación inmediata del álgebra de intervalos clásica es la obtención del rango de valores posible de una función intervalar. En primer lugar, vamos a definir que se entiende por **función intervalar** o **extensión intervalar de una función**:

- **Extensión Natural**

"Sea $f : R^n \rightarrow R$ una función definida a partir de una expresión o algoritmo que implica las cuatro operaciones aritméticas elementales, entonces una **extensión natural** de f, cuyo rango de valores sobre el intervalo X se indica mediante f(X), se obtiene reemplazando cada ocurrencia de x por el intervalo X y realizando todas las operaciones entre intervalos de acuerdo con las definiciones operacionales que se han introducido en el apartado anterior".

Se puede demostrar que:

"Si F(X) es la extensión intervalar natural de una función f(x), y X es un intervalo real contenido dentro del dominio de la función f(x), entonces F(X) contiene el rango de f(x), R[f(x)] para $x \in X$, es decir

$$R[f(x)] = \{f(x) | x \in X\} \subseteq F(X) \qquad (9.2)$$

Dicha propiedad de la extensión intervalar de una función se denomina **propiedad de inclusión**".

Si bien la aritmética de intervalos clásica al ser aplicada en la determinación del rango de las operaciones aritméticas reales obtiene resultados correctos, no sucede los mismo, en general, cuando se aplica a la determinación del rango de funciones a partir de su extensión intervalar.

Por ejemplo, si se desea evaluar el rango de la función

$$f(x) = x^2 - x$$

en el intervalo X=[0,1] utilizando la aritmética intervalar aplicada directamente sobre la extensión intervalar de la función f(x) se obtendrá

$$[0,1]^2 + [0,1] = [0,1] - [0,1] = [-1,1]$$

cuando el rango de f(x) sobre dicho intervalo es [-1/4,0]. Por lo tanto, la aplicación directa del álgebra de intervalos sobre la extensión intervalar de funciones sobrestima el rango de la función. Dicha sobrestimación se debe a que la variable x se supone de forma implícita que varía de forma independiente en el término x^2 y en el termino -x. Es fenómeno aparece cuando una variable aparece más de una vez en una función. A este fenómeno se le denomina el **problema de las multiincidencias** o de la **dependencia intervalar**. Debido a este fenómeno expresiones algebraicas equivalentes en la aritmética real cuando son evaluadas utilizando la aritmética intervalar producen resultados diferentes. Por ejemplo, si la función presentada anteriormente se rescribe como

$$f(x) = x(x-1)$$

entonces al aplicar de nuevo el álgebra de intervalos para evaluar el rango de la función se obtendrá

$$[0,1]([0,1]-1) = [0,1][-10] = [-1,0] \neq [-1,1]$$

Sin embargo, aunque diferentes expresiones equivalentes de una misma función al ser evaluadas utilizando extensiones intervalares y la aritmética intervalar produzcan resultados diferentes, se puede demostrar que la aplicación de la aritmética intervalar sobre la extensión intervalar de una función proporciona una cota del rango real de la función.

Una medida de la calidad de la extensión intervalar F(X) de una función f(x) es el **exceso de tamaño** en la determinación del rango que introduce, exceso que se mide de la siguiente manera

$$w(F(X) - w(R[f(X)]) \quad (9.3)$$

medida introducida por Moore.

Una medida del decrecimiento asintótico del exceso de tamaño cuando el tamaño w(X) decrece se denomina **orden de convergencia** de F, y es debida también a Moore. Así, la extensión intervalar de una función F es de orden $\alpha>0$ si

$$w(F(X) - w(R[f(X)]) = O(w(X)^{\alpha}) \quad (9.4)$$

sea, si existe una constante $c\geq 0$ tal que

$$w(F(X) - w(R[f(X)]) \leq cw(X)^{\alpha} \quad (9.5)$$

De cara a obtener resultados computacionales rápidos es importancia escoger extensiones intervalares de funciones de orden α tan grande como sea posible cuando w(X) se hace pequeño.

Un concepto parecido, aunque independiente del orden, es la idea de **función de Lipschitz**. Una función extensión intervalar se denomina de Lipschitz si existe un número real K, denominada **constante de Lipschitz**, tal que

$$w(F(X)) \leq Kw(X) \quad (9.6)$$

- **"Centered Forms" (Formas Centradas)**

Las "centered forms" son extensiones intervalares de funciones con características especiales introducidas por Moore. Las "centered forms" más utilizadas con las "meanvalue form" o forma del valor medio y la "Taylor form" o forma de Taylor.

"Sea $f : R^n \rightarrow R$ una función diferenciable y sea F'(X) la extensión intervalar del gradiente f' de la función f, entonces se define

$$T_1(X) = f(c) + (X-c)^T F'(X) \quad (9.7)$$

donde c es el punto medio de X, o cualquier otro punto de X, como la extensión intervalar en "meanvalue form" de la función f, o también, conocida como **extensión del valor medio**". Normalmente F'(X) se puede calcular a través de la extensión intervalar natural de f'(x), o bien, mediante diferenciación automática.

Se puede demostrar que si la extensión intervalar de f', F' es Lipschitz, entonces la extensión intervalar de la función f, T_1 tiene un orden de convergencia igual a 2. Dicha demostración puede ser encontrada en [Ratsckhek84].

"Sea $f : R^n \rightarrow R$ una función dos veces diferenciable y sea F''(X) la extensión intervalar de la matriz Hessiana f'' de la función f, entonces se define

$$T_2(X) = f(c) + (X-c)^T f'(c) + \tfrac{1}{2}(X-c)^T F''(X)(X-c) \quad (9.8)$$

donde c es el punto medio de X o cualquier otro punto de X, como la extensión intervalar en "Taylor form", o también, conocida como **extensión de Taylor** de la función f".

Se puede demostrar que si f es dos veces diferenciable y f'' tiene una extensión intervalar F'' acotada, entonces la extensión intervalar de Taylor de f, T_2 tiene un orden de convergencia igual a 2.

9.3 Evaluación Local de Funciones Intervalares: librería LIA InC++

Tal como se ha visto en el apartado anterior, la aplicación del álgebra de intervalos clásica junto con la idea de la extensión intervalar de una función permite evaluar el rango de una función. Esta técnica de evaluación es local puesto que no tiene en cuenta las dependencias globales entre las diferentes instancias de las variables en la expresión. Existen diferentes implementaciones de la álgebra de intervalos clásica sobre diferentes lenguajes de programación estándar, entre ellos:

- Triplex de Cole y Morrison
- Pascal-SC de Bohelnder
- Fortran-SC de Kulish
- IP++ de Alander
- PBasic de Aberth
- VPI de Ely
- LIA InC++ de Hyvönen y De Pascale

- **La implementación del Algebra de Intervalos escogida: LIA InC++**

En concreto nos centraremos en esta última implementación del álgebra de intervalos clásica. Dicha librería permite evaluar el rango de una función aplicando el álgebra de intervalos clásica sobre su extensión intervalar. Utilizando C++, implementa un objeto denominado *Interval* que implementa un intervalo. Sobre dicho objeto se definen las operaciones del álgebra de intervalos.

LIA InC++ además añade una serie de extensiones al álgebra de intervalos clásica que le permiten:

1. Intervalos abiertos

Las implementaciones tradicionales del álgebra de intervalos consideran sólo intervalos cerrados. Ello se debe probablemente a los problemas que enunciamos a continuación que hacen que el significado de un intervalo abierto quede poco claro desde el punto de vista matemático:

(a) Los errores de redondeo debidos a la precisión del ordenador que se está utilizando corrompen las precisión de los límites de los intervalos. Mediante técnicas de redondeo externas normalmente sólo se puede calcular la cota exterior del resultado real.

(b) La existencia de múltiples instancias en la expresión intervalar que se está evaluando normalmente produce un intervalo resultante de mayor anchura que la real, lo que hace que la información sobre el extremo abierto del intervalo resulte poco útil.

Sin embargo, desde el punto de vista práctico, la información sobre el límite de un intervalo es abierto o cerrado es a menudo importante debido a que los límites suelen ser los puntos más importantes de un intervalo. LIA InC++ permite trabajar con intervalos abiertos y semiabiertos del tipo (x,y), [x,y) y (x,y]. El primer problema que se ha presentado se resuelve desde el punto de vista práctico representado el resultado con una precisión menor que la que se ha utilizado para realizar los cálculos. Por otro lado, el segundo problema que se ha presentado es simplemente aceptado como algo inevitable, si la expresión intervalar que se está evaluando posee múltiples instancias de una misma variable la utilidad del resultado obtenido disminuye, tanto si se utilizan intervalos abiertos como cerrados.

2. La noción de infinito

La noción de infinito es implementada: intervalos con límites cuyo valor es $-\infty$, o bien, $+\infty$ son tratados adecuadamente extendiendo las reglas del álgebra de intervalos clásica para que funcionen adecuadamente frente a dichos casos.

3. Intervalos complemento

LIA InC++ implementa la idea de intervalos complemento y generaliza las reglas del álgebra de intervalos para poder operar con ellos. Un intervalo complemento significa "cualquier valor que no este contenido en el rango indicado mediante los extremos del intervalo " y se representa con los delimitadores usuales de un intervalo [ ] pero escritos al revés] [. Utilizando intervalos complemento, las reglas del álgebra de intervalos clásica se pueden extender para determinadas situaciones discontinuas no definidas en dicha álgebra, tales como la división de un intervalo que contienen el cero.

4. Intervalos no conexos

LIA InC++ implementa también extensiones del álgebra de intervalos para poder operar con intervalos no conexos. Un intervalo no conexo es un rango discontinuo de valores que se representa mediante un conjunto de intervalos y de intervalos complemento. Por ejemplo:

$$\{ (1,2),]-4,7(\}$$

La mayor limitación del álgebra de intervalos clásica y por tanto de la implementación de la misma LIA InC++ es que el rango de una función obtenido mediante la su aplicación será sobrestimado si la expresión de la función contiene múltiples instancias. Para evitar este problema se debe de utilizar, lo que se denomina el álgebra de intervalos global, o bien, la aplicación del álgebra de intervalos a la optimización global.

9.4 Aplicación del Álgebra de Intervalos a la Optimización Global

La mayoría de métodos de optimización global tienen al menos uno de los dos defectos siguientes. El primer defecto es que el método puede no ser capaz de garantizar que se hayan encontrado los óptimos globales con una cierta tolerancia (algoritmos estocásticos). Ello implica que los resultados obtenidos están sujetos a cierta duda sobre su validez. El segundo defecto es que el método sólo puede ser aplicable bajo condiciones muy restrictivas tales como el conocimiento de una constante de Lipschitz, convexidad, separabilidad, etc. (este es el caso de la mayoría de algoritmos deterministas). Finalmente, si se intentan resolver los problemas de optimización global utilizando técnicas clásicas basadas en técnicas de descenso es muy probable que se llegue a óptimo local en lugar de un a un óptimo global.

Estos problemas son inherentes a la dificultad de resolver el problema de optimización global, lo que lleva a menudo a mucha gente a creer que la optimización global es un tema intratable.

Los problemas anteriormente comentados se pueden evitar utilizando herramientas y técnicas basadas en el Álgebra de Intervalos en combinación con algoritmos de Branch and Bound. Dentro de este marco se pueden determinar los óptimos globales para una amplia gama de funciones.

Las ideas básicas detrás de tales algoritmos para optimización sin restricciones se pueden expresar con una notación mínima y sin ninguna referencia al álgebra de intervalos. Dichos principios incluyen la comprobación sobre el rango de la función objetivo así como un test computacional de unicidad y existencia.

El problema de optimización global a resolver se puede reformular de la siguiente manera: sea ϕ una función de $R^n \rightarrow R$ siendo $X \in R^n$ el rango valores de la función para el cual se desea determinar el óptimo y sean $C(X) = (c_1(X), \cdots, c_m(X))^T : R^n \rightarrow R^m$ las restricciones del problema de optimización, entonces los algoritmos de Branch and Bound que vamos a presentar en este capítulo permiten determinar de forma rigurosa las cotas inferior y superior de la función objetivo ϕ dentro del problema de optimización siguiente:

$$\begin{aligned} &\min \quad \phi(X) \\ &\text{t.q.:} \quad c_i(X) = 0 \quad i = 1, \cdots, m \\ &\qquad x_{i_j}^- \le x_{i_j} \quad i = 1, \cdots, n \quad j = 1, \cdots, q-\mu \\ &\qquad x_{i_j} \ge x_{i_j}^+ \quad i = 1, \cdots, n \quad j = q-\mu+1, \cdots, q \end{aligned} \tag{9.9}$$

y para cada $X^* \in X$ que minimiza ϕ, permiten determinar las cotas $a_i \le x_i^* \le b_i$ tales que $b_i - a_i$ se puede hacer tan pequeño como se desee y aseguran matemáticamente pero a la vez automáticamente que existe un único punto crítico dentro de cada: $\breve{X} = \{X = (x_1, \cdots, x_n) | a_i \le x_i \le b_i, 1 \le i \le n\}$. Las restricciones de tipo desigualdad

$$g_j(X) \le 0, \quad 1 \le j \le \mu \tag{9.10}$$

se pueden introducir en el problema anterior utilizando variables de holgura

$$c_{m+j}(X) = g_j(X) + x_{n+j}, \quad x_{n+j} \ge 0 \tag{9.11}$$

Además, la caja inicial $X=(x_1, \ldots, x_n)$ define una limitación artificial en la región de búsqueda, mientras que:

$$x_{i_j}^- \le x_{i_j} \le x_{i_j}^+ \tag{9.12}$$

representan restricciones de tipo cota para un subconjunto de variables. Normalmente, como veremos más adelante, si existen restricciones de tipo cota resulta ventajoso tratarlas de forma separada de las restricciones de tipo igualdad y desigualdad.

Sea X el vector intervalar o caja inicial de búsqueda, los algoritmos de Branch and Bound mantienen una o más de las siguientes listas: una lista L de cajas a procesar, una lista U de cajas de tamaño reducido y una lista C de cajas para las cuales se ha verificado que contienen puntos críticos.

El patrón que siguen dichos algoritmos es el siguiente:

Algoritmo 1: Branch and Bound

Entrada: una caja inicial X_o
Salida: una lista C de cajas sobre las que se ha provado que contienen puntos críticos y una lista U de cajas para las cuales la función objetivo toma valores pequeños pero que no han podido ser eliminadas.

Paso 1. Inicialización de la lista L colocando en la misma la región inicial de búsqueda X_o.
Paso 2. Mientras $L \neq \emptyset$:

(a) Extraer la primera caja X de la lista L

(b) Procesar X hasta llegar a:

- Eliminar X
- Reducir el tamaño de X
- Determinar que X contiene un único punto crítico, para entonces determinarlo con un alto grado de precisión
- Subdividir X para que sea más fácil poder eliminar, reducir o verificar la unicidad de las subcajas así obtenidas

(c) Insertar una o más cajas obtenidas a partir de X en las listas L, U y C, dependiendo del tamaño de las subcajas resultantes obtenidas a partir del paso 2(b) y de si el test computacional de existencia en este paso ha determinado un único punto crítico

Muchos detalles, tales como criterios de parada y tolerancias, no se detallan en el Algoritmo 1 que acabamos de presentar. Dichos detalles difieren para cada uno de los algoritmos. Una combinación de diversas técnicas son utilizados por los diferentes solvers de optimización global basada en intervalos para realizar el paso 2b del Algoritmo 1. A continuación se presenta la estructura genérica del algoritmo utilizado por dichos algoritmos para la determinación del rango de la función objetivo así como para la verificación de los puntos críticos.

Algoritmo 2: Determinación del Rango y Verificación de los Puntos Críticos

Entrada: una caja X y una cota superior determinada de forma rigurosa $\overline{\phi}$ del mínimo global.
Salida: una o más cajas obtenidas a partir de X, o bien, la información de que X no puede contener el mínimo global, o bien, la información de que X contiene un punto crítico.

Paso 1. Comprobación de la Factibilidad: (sólo para problemas con restricciones)
(a) Abandonar el algoritmo si se demuestra que X no es factible. Para i = 1 hasta m:
- Calcular una cota de para el rango de cada una las restricciones c_i (X) sobre X.
- Si $0 \notin c_i(X)$ entonces descartamos X y abandonamos del algoritmo.

(b) Probar computacionalmente, si es posible, que exista al menos un punto factible en X.

Paso 2. Comprobación del Rango o "Test del Punto Central"
(a) Calcular una cota inferior $\underline{\phi}(X)$ del rango de ϕ sobre X.
(b) Si dicha cota es superior a la cota superior del rango de la función objetivo disponible hasta el momento $\underline{\phi}(X) > \overline{\phi}$ entonces descartamos X y abandonamos el algoritmo.

Paso 3. Actualización de la Cota Superior del Mínimo
Si el problema es sin restricciones o su factibilidad ha sido probada en el paso 1b, entonces:
(a) Utilizar el álgebra de intervalos para calcular una cota superior de la función obbjetivo $\overline{\phi}(X)$ sobre la caja X.
(b) Actualización de la cota superior de la función objetivo a partir del resultado obtenido en el apartado anterior: $\overline{\phi} \leftarrow \min\{\overline{\phi}, \overline{\phi}(X)\}$

Paso 4. "Test de Monotonicidad"
(a) Calcular una acotación del grandiente de la función objetivo $\nabla\phi(X)$ del rango de $\nabla\phi$ sobre X. (Debe notarse que si X es más fina, es decir, si alguna de las restricciones tipo cota están activas sobre X, entonces se debe utilizar un gradiente reducido)
(b) Si $0 \notin \nabla\phi(X)$ entonces descartamos X y abandonamos el algoritmo.

Paso 5. "Test de Convexidad"
Si la matriz Hessiana $\nabla^2\phi$ no es positiva definida sobre X entonces eliminamos X y abandonamos el algoritmo.

Paso 6. "Convergencia Cuadrática y Existencia Computacional: Unicidad"
Utilizar el método intervalar de Newton (con las condiciones de Fritz-John para el caso de que existan restricciones) para realizar uno o más de los siguientes pasos:
- Reducir el tamaño de X
- Eliminar X
- Verificar que existe un único punto crítico en X.

Paso 7. "Bisección o Teselación Geométrica"
Si el paso 6 no se traduce en un cambio significante en X, entonces biseccionaremos X a lo largo de la dirección de una coordenada (o bien, en caso contrario teselaremos X), devolviendo todas las cajas resultantes para procesarlas posteriormente.

9.5 Aplicación a Problemas de Optimización Global con Restricciones

- **Verificación de la Factibilidad**

Con las restricciones deben realizarse los siguientes cálculos:

(a) Utilizar las restricciones para eliminar partes de la región X que no sean factibles
(b) Probar la factibilidad respecto a las restricciones desigualdad
(c) Probar la factibilidad respecto a las restricciones igualdad

Si las restricciones desigualdad se convierten en primer lugar en restricciones igualdad, las técnicas de precondicionado junto con el método intervalar de Gauss-Seidel, o bien, con el método intervalar de eliminación de Gauss pueden ser utilizadas directamente sobre el sistema de $m \times n$ restricciones.

- **Restricciones igualdad y desigualdad**

A primera vista, los problemas con restricciones desigualdad parecen más fáciles que las restricciones igualdad. Esto se debe a que la factibilidad puede ser probada, a veces, sin utilizar el método de Newton intervalar: simplemente acotando el rango de cada restricción desigualdad d_j utilizando aritmética intervalar, y a continuación comprobar que las cotas superiores obtenidas son todas negativas, es decir, comprobando que $d_j(X) \leq 0$ para cada j.

- **Restricciones tipo cota**

Las restricciones tipo cota, $a_{ij} \leq x_{ij} \leq b_{ij}$ son tratadas descomponiendo la región en todas las posibles subregiones de dimensiones menores, tal como se ilustra en la Fig. 9.1 para n = 2. Esta subregiones se colocan en la lista L y son procesadas de la forma usual, excepto que se utilizan los gradientes y matrices Hessianas reducidas en el método de Newton intervalar sobre las regiones de menor dimensión. Si todas las cajas se almacenan en L, no es difícil prever que el número total de cajas de todas las dimensiones así obtenidas será 3^p, donde p es el número de restricciones de tipo cota. Sin embargo, si se dispone de una buena cota superior $\overline{\phi}$ del mínimo global, entonces muchas de las cajas pueden ser eliminadas durante el proceso de "peeling".

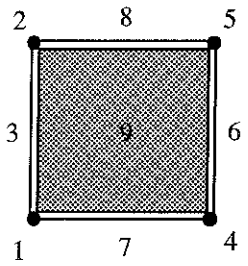


Fig. 9.1 Proceso de "peeling"

La estructura del algoritmo que proporciona la lista de cajas de dimensión inferior se puede describir en forma recursiva de la siguiente manera:

Algoritmo 3: "Peeling" the Boundary

Paso 1. Tomamos el índice de coordenadas actual i, la caja $X = (x_1, \cdots, x_n)$, la lista I de los índices de las coordenadas de la caja X que ya se han considerado, y la lista L de las cajas que ya se han generado para ser procesadas posteriormente.

Paso 2. Si $i \notin I$ entonces:
(a) *Procesamos la caja de cotas inferiores*:

- Fijamos todas las coordenadas de $\tilde{X}$ excepto la i-ésima a las correspondientes coordenadas de X. Fijamos la i-ésima coordenada de $\tilde{X}$ a $\underline{x}_i$ (cota inferior de dicha coordenada).
- Creamos una nueva lista de índices I_{new} a partir de la lista anterior I y del índice en curso i.
- Si $i = n$ entonces almacenamos $\tilde{X}$ en la lista L, sino ejecutamos el algoritmo con $i+1$, $\tilde{X}$, y I_{new} reemplazando i, X y I, respectivamente.

(b) *Procesamos la caja de cotas superiores*:

- Fijamos todas las coordenadas de $\tilde{X}$ excepto la i-ésima a las correspondientes coordenadas de X. Fijamos la i-ésima coordenada de $\tilde{X}$ a $\overline{x}_i$ (cota superior de dicha coordenada).
- Creamos una nueva lista de índices I_{new} a partir de la lista anterior I y del índice en curso i.
- Si $i = n$ entonces almacenamos $\tilde{X}$ en la lista L, sino ejecutamos el algoritmo con $i+1$, $\tilde{X}$, y I_{new} reemplazando i, X y I, respectivamente.

(c) *Procesamos la caja interior*:

- Fijamos $\tilde{X} = X$.
- Creamos una nueva lista de índices I_{new} a partir de la lista anterior I y del índice en curso i.
- Si $i = n$ entonces almacenamos $\tilde{X}$ en la lista L, sino ejecutamos el algoritmo con $i+1$, $\tilde{X}$, y I_{new} reemplazando i, X y I, respectivamente.

La numeración de las nueve cajas de dimensiones 2,1 y 0 de la Fig. 9.1 representa el orden en que aparecerían en la lista L si cada caja generada con i-n, se almacenara en los pasos 2(a)iii, 2(b)iii y 2(c)iii del Algortimo 3. El orden de procesado del Algoritmo 3 se puede ver como un recorrido a través de un árbol ternario, como en el de la Fig. 9.2. Los niveles correspondientes a este árbol corresponden a las coordenadas i, con la raíz en i = 1 en el nivel superior y las hojas en i = n en el nivel superior. Tal como se presenta en la Fig. 9.2, el orden en que las hojas aparecen en L es de izquierda a derecha.

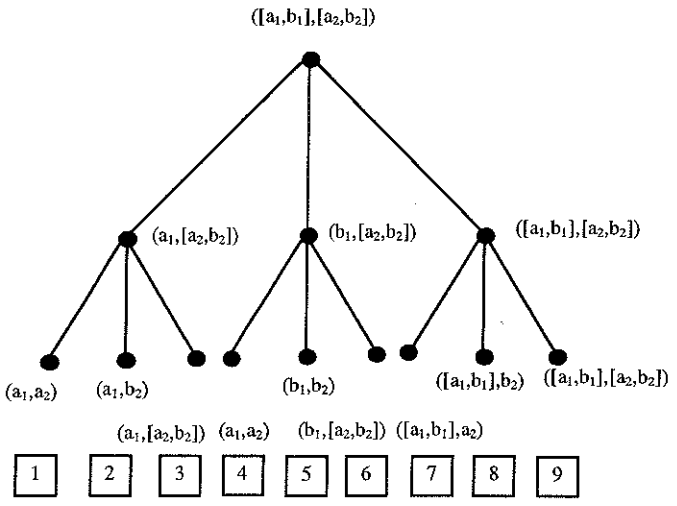


Fig. 9.2 Arbol asociado al proceso de "peeling"

Por supuesto, la implementación del Algoritmo 3 tiene los siguientes pasos adicionales:

- Eliminar las cajas X y $\tilde{X}$ antes de realizar un procesado posteriores o almacenamiento en L comprobando $\phi(X)$ o $\phi(\tilde{X})$ y el gradiente reducido de ϕ sobre X o $\tilde{X}$.
- No considerar las coordenadas i para las cuáles las cotas $a_i \leq x_i \leq b_i$ representan la extensión de la región de búsqueda, y no restricciones reales del problema.

Estos pasos no se han introducido en la presentación del Algoritmo 3 por claridad. Sin embargo, pueden resultar indispensables para reducir el número de cajas en L a un número práctico. Debe observarse que dichos pasos pueden podar el árbol de la Fig. 9.2 en el nivel más alto.

Ratz [Ratz94] propone una técnica alternativa para evitar la posibles expansión exponencial en el número de cajas en la lista L cuando se aplica el algoritmo de "peeling" anterior. En el algoritmo de Ratz, partes de los lados se ponen en la lista L cuando un método de Newton intervalar reducido después de que rechace segmentos como posibles soluciones. El algoritmo de Ratz sigue lo siguientes pasos:

Algoritmo 4: Algoritmo de Ratz para restricciones tipo cota

El punto de partida del algoritmo es una caja X, con p de cuyas coordenadas $x_{i_1}, \cdots, x_{i_p}$ se corresponden a intervalos, y el resto de coordenadas corresponden a restricciones tipo cotas activas.

Para j = 1 a p:

Paso 1. Realizar un paso del método de Gauss-Seidel intervalar para la i_j coordenada x_{i_j} de X, obteniendo $\overline{x}_{i_j}$

Paso 2. Si $\underline{x}_{i_j}$ corresponde a una restricción tipo cota activa y $\overline{x}_{i_j} > \underline{x}_{i_j}$ entonces:
(a) Construir una caja X_{bdy} cuya i-ésima coordenada sea $\overline{x}_i$, para $i \neq i_j$ y cuya i_j -ésima coordenada sea $\underline{x}_{i_j}$.
(b) Colocar X_{bdy} en la lista L.

Paso 3. Si $\overline{x}_{i_j}$ corresponde a una restricción tipo cota activa y $\overline{x}_{i_j} > \overline{x}_{i_j}$ entonces:
(a) Construir una caja X_{bdy} cuya i-ésima coordenada sea $\overline{x}_i$, para $i \neq i_j$ y cuya i_j -ésima coordenada sea $\overline{x}_{i_j}$.
(b) Colocar X_{bdy} en la lista L.

La estructura de un algoritmo de optimización de problemas con restricciones tipo cota que incluyera el algoritmo de Ratz que acabamos de describir sería idéntico que el algoritmo de optimización de problemas sin restricciones. La única diferencia consistiría que en el paso de Newton intervalar sería reemplazado por el algoritmo de Ratz. Dicho algoritmo produce menos cajas que el algoritmo de "peeling", pero su utilidad debe aún ser verificada empíricamente. Concretamente, el algoritmo de "peeling" evalúa la función objetivo en muchas partes de dimensión pequeña de los lados de la caja inicial muy pronto en el proceso de branch and bound: esto se traduce en una mejor estimación del óptimo global y por lo tanto en una eliminación más rápida de la cajas que no contienen el óptimo.

9.6 Utilización de los Métodos de Newton Intervalares

Los métodos de Newton Intervalares se utilizan en el análisis de los sistemas algebraicos no lineales que aparecen durante la resolución de problemas de optimización global para:

(a) Reducir el tamaño de las regiones X, con una convergencia cuadrática del tamaño de las mismas hacia cero.
(b) Probar computacionalmente (pero rigurosamente) la existencia de soluciones dentro de una región X
(c) Probar que existe una única solución dentro de la región X
(d) Probar que no puede existir ninguna solución dentro de la región X

9.6.1 Descripción del método de Newton intervalar

Dada una función f, podemos aproximar el valor de la misma en un punto por

$$f(y) \in f(x)+J(x,X)(y-x) \qquad (9.13)$$

donde J(x,X) es el Jacobiano de la función f.

Si y es un cero de la función, entonces f(y)=0, de forma que

$$0 \in f(x)+J(x,X)(y-x) \qquad (9.14)$$

Entonces, se define el conjunto solución de dicha ecuación como:

$$S=\{y: f(x)+J(x,x')(y-x)=0\} \qquad (9.15)$$

para todo $x' \in X$. El conjunto S contiene cualquier punto $y \in X$ para el cual f(y)=0. Cuanto menor sea la caja X, menor será el conjunto S. El objetivo del método de Newton intervalar es reducir el conjunto X hasta que el conjunto S sea tan pequeño como se desee, de forma que un punto solución $y \in X$ este acotado con precisión. Debe observarse que en general S no será una caja.

Para determinar el conjunto solución, escribiremos la ecuación a resolver de la siguiente manera:

$$f(x)+J(x,X)(Y-x)=0 \qquad (9.16)$$

de forma que obtendremos una caja Y que contendrá al conjunto solución S.

Para enfatizar la dependencia de Y con X y x utilizaremos N(x,X) en lugar de Y, de forma que definiremos el siguiente algoritmo iterativo para resolver la ecuación anterior

$$\begin{aligned} &f(x_k)+J(x_k,X_k)[N(x_k,X_k)-x_k]=0 \\ &X_{k+1}=X_k \cap N(x_k,X_k) \end{aligned} \qquad (9.17)$$

para k=0,1,2,... donde x_k debe ser un punto de X_k. Una buena elección para x_k es el centro de X_k, $m(X_k)$.

El primer paso para resolver la primera de las ecuaciones del algoritmo iterativo es multiplicar dicha ecuación por una inversa aproximada B del cento J^c de la matriz J(x,X). Cuando J^c es singular, se debe subdividir la caja X en dos y trabajar con cada caja por separado. De esta forma, se obtendrán matrices J^c diferentes para las nuevas cajas que probablemente no serán singulares.

Si definimos:

$$\begin{aligned} &M(x,X)=BJ(x,X) \\ &r=-Bf(x) \end{aligned} \qquad (9.18)$$

el algoritmo iterativo anterior, se puede escribir de la siguiente manera

$$M(x_k, X_k)[N(x_k, X_k) - x_k] = r$$
$$X_{k+1} = X_k \cap N(x_k, X_k) \qquad (9.19)$$

Normalmente, nos referimos al método de Newton intervalar en el cual se resuelve la primera ecuación del algoritmo iterativo anterior mediante un paso del método de Gauss-Seidel. Entonces, la iteración anterior se puede escribir de la siguiente manera

$$N_i = x_i + (M_{ii})^{-1}\left[r_i - \sum_{j=1}^{i-1} M_{ij}(X_j' - x_j) - \sum_{j=i+1}^{n} M_{ij}(X_j - x_j) \right] \qquad (9.20)$$
$$X_i' = N_i \cap X_i$$

A continuación vamos a describir los pasos del método de Newton intervalar propuesto por Hansen. El algoritmo supone que la caja de búsqueda inicial $X^{(0)}$ viene dada. A medida que algoritmo opera con dicha caja, va generando subcajas a partir de ella. Estas subcajas se almacenan en una lista L mientras esperan a ser procesadas. La caja que está procesando actualmente el algoritmo se elimina de dicha lista cuando es escogida para ser procesada. Los pasos del algoritmo son los siguientes:

Algoritmo 5: Algoritmo de Newton Intervalar

Paso 0. Inicialización del algoritmo colocando la caja inicial $X^{(0)}$ en la lista L.

Paso 1. Si la lista L está vacía, el algoritmo se detiene. En caso contrario, se escoge la caja que más recientemente se ha añadido a la lista como caja a procesar. A dicha caja la denominaremos X. Una vez seleccionada se elimina de la lista L. Utilizando la aritmética de redondeo, calculamos una aproximación x para el centro de la caja m(X).

Paso 2. Cálculo de J(x,X) utilizando una expansión basada en

$$f(y) \in f(x) + \sum_{i=1}^{n} (y_i - x_i) g_i(X_1, \cdots, X_i, x_{i+1}, \cdots, x_n)$$

sea, utilizando una expansión que tenga tantos argumentos reales como sea posible. Una alternativa mejor es calcular la función matriz de la pendiente G(x,X) en lugar de J(x,X). Una vez determinada la función J(x,X), se calculará una aproximación J^c para el centro de J(x,X). Para a continuación calcular una inversa aproximada B de J^c. Finalmente, se calculará M(x,X)=BJ(x,X), f(x) y r(x)=-Bf(x).

Paso 3. Si M(x,X) es diagonal dominante, entonces saltaremos al paso 12.

Paso 4. Aplicaremos un paso del método de Gauss-Seidel para resolver M(x,X)[N(x,X)-x]=r(x). Durante el proceso, se realizará la intersección X' = X ∩ N. El conjunto resultante X' puede resultar vacío, una única caja, o una caja con agujeros en una o más componentes. Si X' no está vacío, denominaremos Y' a la caja más pequeña que contiene X'. Por lo tanto, Y'=X' si X' es una única caja. En caso contrario, Y' es igual a X' excepto que no tiene agujeros.

Paso 5. Si X' esta vacío, entonces saltaremos al paso 1.

Paso 6. Si Y' satisface $w(Y') < \varepsilon_{Y'}$ y $\|f(X)\| < \varepsilon_F$, entonces mostramos Y' y saltamos al paso 1.

Paso 7. Si X' contiene un agujero en una de sus componentes, de forma que si dicha componente viene representada por el intervalo [a,b] y el agujero viene representado por el intervalo (b,c), la eliminación de dicho agujero comportará la creación de dos subintervalos [a,b] y [c,d]. El agujero se considerará adecuado dividirlo si se cumple $\min\{c-a, d-b\} \geq 0.25 w(X')$, de forma que saltamos al paso 12.

Paso 8. Comprobamos si el algoritmo ha conseguido progresar suficientemente. Para ello, evaluaremos D de acuerdo con

$$D+\frac{n+47}{5n+88}w(X)-\max\{w(X_i)-w(X_i')\}$$

utilizando Y' en lugar de X'. Si $D \leq 0$, entonces fijaremos el flag F=1, renombraremos Y' como X y saltaremos al paso 4. Si D>0 y el flag F=1, resetearemos F haciendo F=0, colocaremos Y' en la lista L y saltaremos al paso 1.

Paso 9. Si Y'=X, saltaremos al paso 10. En caso contrario, utilizaremos

$$T_r \geq w(X_j')\sum_{i=1}^{n}\left|J_{ij}(x,X)\right|$$

para seleccionar la componente de Y' a subdividir. Entonces subdividiremos Y' por la mitad utilizando la componente seleccionada, y colocaremos las dos mitades en la lista L, y saltaremos al paso 1.

Paso 10. Utilizaremos

$$T_r \geq w(X_j')\sum_{i=1}^{n}\left|J_{ij}(x,X)\right|$$

para seleccionar los tres, o menos si existen menos, mejores componentes de Y' para subdividir. Si n=2 seleccionaremos sólo dos, o uno si sólo existe uno. Entonces subdividiremos Y' por la mitad cada de una de las componentes seleccionadas, colocando las subcajas así generadas por las subdivisiones en la lista L, saltando al paso 1.

Paso 11. Empezaremos por el mayor agujero determinado por $\min\{c-a,d-b\} \geq 0.25w(X')$ como el más adecuado para realizar la subdivisión, subdividiendo Y' por dicho agujero. Continuaremos hasta que se hayan realizado tres subdivisiones o no haya más agujeros adecuados para ser subdivididos. A continuación colocaremos las subcajas así generadas en la lista L, y saltaremos al paso 1.

Paso 12. Trataremos de descomponer M(x,X) en un producto $L^I U^I$ de factores triangulares, sin realizar intercambios para realizar pivotes. Si la descomposición no se puede completar debido a que la división por un intervalo que contiene el cero, saltaremos al paso15.

Paso 13. Resolveremos $L^I U^I(y-x)=r(x)$. Una vez determinada cada una de las componentes de N(x,X), aplicaremos la intersección $X^{(k+1)}=X^{(k)}\cap N(x^{(k)},X^{(k)})$, denominando al resultado X'. Si X' es vacío, entonces saltaremos al paso 1.

Paso 14. Si se satisface el siguiente criterio: $0 \in f^I(x)$, entonces M(x,X) es diagonal dominante y $N(x,X) \supset X$, mostraremos X' y saltaremos al paso 1. En caso contrario, saltaremos al paso 16.

Paso 15. Resolveremos M(x,X)(y-x)=r(x) para y utilizando el método de Gauss-Seidel, para a continuación realizar la intersección $X^{(k+1)}=X^{(k)}\cap N(x^{(k)},X^{(k)})$, denominado al resultado X'. Si X' es vacio, saltaremos al paso 1. En caso contrario, saltaremos al paso 14.

Paso 16. Comprobaremos si el algoritmo ha realizado suficiente progreso en el paso 13 evaluando D mediante

$$D+\frac{n+47}{5n+88}w(X)-\max\{w(X_i)-w(X_i')\}$$

Si $D \leq 0$, saltaremos al paso 17, en caso contrario saltaremos al paso 23.

Paso 17. Calcularemos J(X) donde J(X) es independiente de x. O sea, todos los argumentos de todos los elementos de J son intervalos. Calcularemos una aproximación para $J^c = m(X)$ y una inversa aproximada B de J^c, para a continuación calcular M(X) = BJ(X).

Paso 18. Empezando en el punto m(X), realizaremos la iteración $z^{(i+1)} = z^{(i)} - Bf(z^{(i)})$ para determinar un cero de f en la caja X. El resultado se indicará mediante x. A continuación calcularemos f(x) y r(x) = Bf(x).

Paso 19. Trataremos de descomponer M(X) en un producto $L^I U^I$ de factores triangulares. Si la descomposición no se puede completar debido a la división de un intervalo que contiene el cero, saltaremos al paso 15.

Paso 20. Resolveremos $L^I U^I (y - x) = r(x)$ para y, para a continuación realizar la intersección $X^{(k+1)} = X^{(k)} \cap N(x^{(k)}, X^{(k)})$, denominando al resultado X'. Si X' está vacío saltaremos al paso 1.

Paso 21. Si se satisface el siguiente criterio: $0 \in f^I(x)$, entonces M(x,X) es diagonal dominante y $N(x, X) \supset X$, mostraremos X' y saltaremos al paso 1.

Paso 22. Comprobaremos si el algoritmo ha progresado suficientemente después de ejecutar el paso 20. Para ello evaluaremos D utilizando

$$D + \frac{n+47}{5n+88} w(X) - \max\{w(X_i) - w(X_i')\}$$

Si D≤0, renombraremos X' como X, fijaremos el flag F'=1, realizaremos el paso 18 y saltaremos al paso 20. Si D>0 y F'=1, resetearemos el flag F', F'=0 y saltaremos al paso 17.

Paso 23. Utilizando $T_r \geq w(X_j') \sum_{i=1}^{n} \left| J_{ij}(x, X) \right|$ seleccionaremos la mejor componente de la caja a dividir, para una vez realizada la selección subdividir la caja por la mitad por dicha componente. Para a continuación colocar las dos mitades en la lista L y saltaremos al paso 1.

9.6.2 Utilización del método de Newton intervalar en el algoritmo de optimización global

Un método de Newton intervalar es un operador N(X,F) asociado a una función F: $R^p \rightarrow R^p$ definida sobre una caja X. Intuitivamente los métodos de Newton intervalares se pueden ver como métodos para acotar el conjunto solución del siguiente sistema lineal intervalar:

$$A(X - x_c) = -F(x_c) \qquad (9.21)$$

El punto x^c es el punto medio de la caja X, una aproximación a un solución de F(X) = 0, o cualquier otro punto que se haya escogido de forma apropiada (normalmente, aunque no siempre, $x_c \in X$). La matriz intervalar A se puede escoger de forma que sea una matriz Lipschitz. En particular, la evaluación de la matriz Jacobiana de F conduce a una matriz A que es Lipschitz, siendo adecuada para la mayoría de los casos.

El método de Newton intervalar no se aplica directamente sobre la ecuación $A(X - x_c) = -F(x_c)$, sino sobre la ecuación precondicionada

$$YA(X - x_c) = -YF(x_c) \quad (9.22)$$

donde Y es una matriz de *p x p* que hace que el conjunto solución de la misma sea más fácil de acotar que el de la ecuación original. Sin embargo, el conjunto solución de la ecuación $A(X - x_c) = -F(x_c)$ es en general un subconjunto de la solución de la ecuación precondicionada.

En la literatura existen diversas formas de acotar el conjunto solución de ambas ecuaciones. Existen tres métodos muy comunes de calcular las cotas de los intervalos del conjunto solución de las mismas. Dichos métodos son: el método de Krawczyk, el método de Gauss-Seidel y el método de eliminación de Gauss intervalar.
Sean Y las cotas obtenidas sobre el conjunto solución de la ecuación $A(X-Y)=-F(Y)$. Entonces, tenemos que:

- Sea A una matriz pendiente para X basada en x_c (o en una matriz Lipschitz sobre X). Si $Y \subset \text{int}(X)$ (o sea, si Y está contenido en el interior topológico de X), entonces existe una solución de F(X) = 0 en X.
- Sea A una matriz pendiente para X basada en x_c (o en una matriz Lipschitz sobre X). Si $Y \cap X = \varnothing$ entonces no existe ninguna solución de F(X) = 0 en X.
- Sea A una matriz Lipschitz sobre X. Si $Y \subset \text{int}(X)$ (o sea, si Y está contenido en el interior topológico de X), entonces existe una única solución de F(X) = 0 sobre X.
- Sea A una matriz pendiente para X basada en x_c (o en una matriz Lipschitz sobre X), entonces cualquier solución de F(X)=0 en X está contenida también en Y. Además, bajo ciertas condiciones, reemplazando X por Y se consigue convergencia cuadrática hacia a cero del tamaño de las componentes de X, permitiendo que las subregiones sean rigurosamente exploradas sin tener que realizar un particionado excesivo.

Por lo tanto, la utilización de un método intervalar de Newton puede tener diferentes objetivos (existencia, unicidad, no existencia, o bien, reducción del tamaño de X. Además, los métodos intervalares de Newton se pueden aplicar a diferentes tipos de sistemas: sistemas no lineales en general, multiplicadores de Lagrange, sistemas Fritz-John, etc. Basado en estas o otras consideraciones, se pueden construir diferentes métodos intervalares de Newton. En resumen, los elementos que se pueden variar son:

(a) Elección del método de solución: Gauss-Seidel, Krawczyk, o bien, eliminación de Gauss intervalar
(b) Elección del precondicionador
(c) Elección de la matriz A (matriz pendiente o matriz Lipschitz, así como su método de cálculo)
(d) Elección del punto de base x_c.

9.7 Algoritmos de Hansen

En este apartado se presentan los diversos algoritmos propuestos por Eldon Hansen [Hansen92] en su libro "Global Optimization using Interval Analysis". Dichos algoritmos constituyen una referencia obligada dentro de los algoritmos de optimización global basados en la búsqueda exhaustiva basados en la técnica de branch and bound en combinación con el álgebra de intervalos clásica. Así mismo los solvers que se han utilizado durante la realización de está tesis para resolver el problema de optimización global planteando al aplicar el nuevo algoritmo de generación de envolventes, se basan también en estos algoritmos. En primer lugar describiremos el algoritmo de Hansen para problemas de optimización sin restricciones para a continuación, presentar como se debe modificar dicho algoritmo para el caso de problemas con restricciones tanto igualdad como desigualdad.

9.7.1 Optimización sin restricciones

El algoritmo de Hansen para resolver problemas de optimización sin restricciones, considera el siguiente problema:

$$\min \text{global } f(x)$$

donde f es una función escalar de un vector x de n componentes. El algoritmo determinará el valor mínimo global f^* de f y los puntos x^* en los cuales dicho mínimo ocurre. Básicamente dicho algoritmo funciona de la siguiente manera:

Paso 1. Inicialización del algoritmo con una caja $X^{(0)}$ en la cual se busca el mínimo global.
Paso 2. Eliminación de las subcajas de $X^{(0)}$ que no pueden contener ningún punto solución, utilizando procedimientos seguros de forma que, a pesar de los errores de redondeo, los puntos donde se alcanza el mínimo global nunca serán borrados.
Paso 3. Iteración del paso 2 hasta conseguir un conjunto de puntos suficientemente pequeño. Puesto que x^* debe encontrarse dentro de este conjunto, su localización está acotada. Entonces acotamos f sobre este conjunto de puntos para obtener las cotas del mínimo global f^*.

Los procedimientos a utilizar el paso 2 son los siguientes:

(a) Borrar las subcajas en las cuales el gradiente g de la función f sea distinto de cero. Esto puede realizarse sin utilizar las derivadas de g tal como se verá a continuación. De forma alternativa, se puede aplicar el método de Newton intervalar para resolver la ecuación g=0, de forma que se puede eliminar las subcajas en las cuales $g \neq 0$.
(b) Muestrear f en varios puntos. El menor valor obtenido de dichos muestreos $\overline{f}$ es una cota superior del mínimo global f^*. Entonces se pueden eliminar las subcajas para las que se cumpla que $f > \overline{f}$.
(c) Eliminar las subcajas en las cuales f no sea convexa.

A continuación se presenta los pasos detallados del algoritmo de Hansen, y después de dicha presentación se comenta cada uno de ellos en profundidad. Así pues el algoritmo de Hansen para problemas de optimización sin restricciones es el siguiente:

Algoritmo 6: Algoritmo de Hansen sin restricciones

Paso 0. Inicialización del algoritmo colocando la caja inicial $X^{(0)}$ en la lista L.

Paso 1. Evaluación de la función objetivo f en el centro de cada una de las cajas iniciales X contenidas en la lista L_1. Utilización del resultado para actualizar $\overline{f}$.

Paso 2. Evaluación de f(X) para cada caja inicial X contenida en L_1. El resultado obtenido se indicará de la siguiente manera: $[f^L(X), f^R(X)]$.

Paso 3. Eliminación de cualquier caja X contenida en L_1 para la que se cumpla $f^L(X) > \overline{f}$.

Paso 4. Si la lista L_1 está vacía, saltamos al Paso 13. En caso contrario, determinamos la caja X contenida en la lista L_1 que tenga el menor valor de $f^L(X)$. Selección de dicha caja como una de las cajas a procesar a continuación por el algoritmo.

Paso 5. Test de monotonicidad: si $0 \notin g_i(X)$ para algún i = 1,...,n entonces descartamos la caja X y volvemos al paso 4.

Paso 6. Test de no convexidad: sea H la matriz Hessiana de la función f y h_{ii} un elemento de la diagonal de dicha matriz, si $h_{ii}(X_1, ..., X_n) < 0$ para cualquier i = 1, ... , n, entonces descartamos la caja X y volvemos al paso 4.

Paso 7. Mediante el método lineal borraremos toda o parte de la caja X.

Paso 8. Mediante el método cuadrático borraremos toda o parte de la caja X.

Paso 9. Aplicación de un paso del método de Newton intervalar. Si el resultado es un conjunto vacío, volvemos al paso 4. Mientras que si el paso realizado con el método de Newton intervalar prueba la existencia de una solución en la caja X, registramos dicha información.

Paso 10. Si $w(X) < \varepsilon_X$, saltamos al paso 12. En caso contrario, mezclamos cualquiera de los espacios existentes en las componentes de X que se hubieran generado en el paso 9 con los espacios generados en pasos anteriores. Si existen todavía cualquier espacio que satisfaga la condición de división, utilizaremos los tres mayores para dividir la caja X en subcajas. Entonces saltaremos al paso 12. En caso de que no quede ningún espacio que cumpla la condición de división, saltaremos al paso 11.

Paso 11. Si X es una versión suficientemente reducida de la versión de la caja que existía en el paso 4, saltaremos al paso 12. En caso contrario, dividiremos la caja X en dos para les tres componentes con mayor D_i, donde $D_i(X)=W[g_i(X)]w(X_i)$ para i=1,...,n.

Paso 12. Para cada caja X resultante de los pasos 10 o 11, evaluaremos f en el centro y utilizaremos el resultado de dicha evaluación para actualizar $\overline{f}$. Además, para cada una de dichas cajas X, evaluaremos w(X) y f(X). Si $w(X) < \varepsilon_X$ y $w[f(X)] < \varepsilon_F$, colocaremos X en la lista L_2. En caso contrario, colocaremos X en la lista L_1. Finalmente saltaremos al paso 4.

Paso 13. Para cada caja X contenida en la lista L_2, habremos evaluado f(X) en el paso 12. El resultado de dicha evaluación se indicará mediante $[f^L(X), f^R(X)]$.
Eliminaremos cualquier caja que cumpla $f^L(X) > \overline{f}$. El conjunto de cajas que han quedado sin eliminar las denominaremos: $X^{(1)}$, ..., $X^{(s)}$ donde s es el número de cajas que han quedado sin eliminar. Determinaremos la cota inferior para el óptimo global de f^* mediante: $\underline{f} = \min_{1 \le i \le s} f^L(X^{(i)})$.
Paso 14. Finalización del algoritmo.

A continuación vamos a explicar con un poco más de detalle cada uno de los pasos del algoritmo de Hansen, para el caso de problemas de optimización sin restricciones.

- **La Caja Inicial** (Paso 0)

Para un problema de optimización sin restricciones, la solución del problema se puede encontrar en cualquier punto a una distancia arbitrariamente lejana del origen. Puesto que en la práctica debemos acotar el espacio de búsqueda, se restringirá dicho espacio a una caja X.

- **La Utilización del Gradiente** (Paso 5)

Supongamos que la función f es continuamente diferenciable. Entonces, el gradiente g de f es cero en el mínimo global, aunque también g es cero en los mínimos y máximos locales y en los puntos de silla. El algoritmo de Hansen utiliza el método intervalar de Newton para determinar y acotar los ceros del gradiente. Sin embargo, computacionalmente es muy costoso utilizar dicho método: se debe evaluar el Jacobiano de g así como varios procedimientos algebraicos. Por lo tanto, el algoritmo de Hansen utiliza como test de monotonicidad un test más simple basado en el gradiente.
Sea X una subcaja de $X^{(0)}$. Sobre dicha caja evaluamos las componentes del gradiente g, o sea, $g_i(X)$ para i = 1, ..., n. El intervalo resultante se indica mediante: $[g_i^L(X), g_i^R(X)]$. Si $g_i^L(X) > 0$, o si, $g_i^R(X) < 0$, entonces g(x) no es cero para cualquier $x \in X$. Por lo tanto, no existe ningún punto estacionario de f en X. En particular, X no puede contener el mínimo global.
Si el test del gradiente no tiene éxito al ser aplicado sobre X, entonces el algoritmo de Hansen aplica el método intervalar de Newton sobre dicho intervalo.

- **La Cota Superior** (Paso 3)

A medida que el algoritmo avanza, se evalúa la función f en varios puntos de $X^{(0)}$. Cada valor de f obtenido de esta forma es una cota superior del valor del mínimo global f^* de f, debiendo hacer uso de la mejor cota obtenida de esta forma.
Así por ejemplo, si se ha calculado que el valor de f en x = 1: f(1) = 2, entonces podemos afirmar que el mínimo global $f^* \le 2$. De forma que si al analizar el intervalo $X = [2, 10^{30}]$, se obtiene $f(X) = [4, 3 \cdot 10^{60}]$. Entonces $f(x) > 2 \ge f^*$ para todo $x \in X$, pudiéndose afirmar que el mínimo global de f no se encuentra en X.

- **La Búsqueda Real** (Paso 7 y 8)

Cuanto más cercano se encuentre la cota superior de $\bar{f}$ del mínimo global f^*, mayor será el número de puntos que se podrán eliminar utilizando el procedimiento descrito en el punto anterior. A continuación vamos a describir un método para decrementar el valor de la cota superior $\bar{f}$.
Cada vez que se genera una nueva caja X, se evalúa la función f en su centro m(X) y se utiliza el resultado para reducir la cota superior $\bar{f}$ siempre que sea posible. Está técnica es una técnica de muestreo bastante simple para determinar x^*, el punto donde se alcanza el mínimo global. Por ello se debe utilizar un procedimiento más sofisticado. Supongamos que la evaluación de f en m(X) consigue reducir la cota superior $\bar{f}$, entonces es muy probable que existan puntos de X en los cuales f toma aún un valor menor, el procedimiento que se propone en este apartado pretende determinar dichos puntos.
Dicho procedimiento consiste en aplicar el método intervalar de Newton en la caja X. Para ello, calcularemos una aproximación de la inversa de B en el centro del Jacobiano intervalar. Empezando en $x_o = m(X)$ definiremos

$$x_{k+1} = x_k - Bg(x_k) \quad \text{para} \quad k = 1,2,\cdots \qquad (9.23)$$

Dicha búsqueda se detendrá si $f(x_{k+1}) \geq f(x_k)$. En dicho caso, el penúltimo punto se utilizará para actualizar la cota superior $\bar{f}$. La búsqueda también se detendrá si x_{k+1} es punto que se encuentra fuera de la caja X. En este caso, determinaremos el punto x en el cual el segmento lineal que une x_k y x_{k+1} cruza la frontera de X, entonces evaluaremos f en dicho punto y utilizaremos el resultado para actualizar $\bar{f}$.
Finalmente un tercer criterio de finalización se aplicará si el paso de actualización se hace demasiado pequeño, es decir, si

$$\left\| x_{k+1} - x_k \right\| < \varepsilon \qquad (9.24)$$

para un determinado $\varepsilon>0$. En este caso, utilizaremos el menor valor de entre $f(x_k)$ y $f(x_{k+1})$ para actualizar $\bar{f}$.

- **El Test de No Convexidad** (Paso 6)

Si f(x) tiene un mínimo en x^*, entonces f debe ser convexa en la vecindad de x^*. Por lo tanto, la Hessiana H(x) de f debe ser semidefinida positiva en x^*. La condición necesaria para que la Hessiana sea semidefinida positiva es que los elementos de la diagonal H_{ii} (i = 1, ... , n) de H sea no negativos.
Consideremos la caja X. Si H_{ii} (X) < 0 para algún i = 1, ... , n, entonces H_{ii} (x) < 0 para todo $x \in X$. Por lo tanto, H no puede ser semidefinida positiva para cualquier punto de X. Por lo tanto, f no puede tener un mínimo en X, pudiéndose eliminar.
Existen otras condiciones suficientes que H debe satisfacer para que sea semidefinida positiva: por ejemplo, los menores principales dominantes de H de todos los órdenes 1, ..., n deben ser no negativos. Podríamos pues comprobar si una o más de estas condiciones se han violado sobre X, y por lo tanto eliminarla. Sin embargo, el esfuerzo extra probablemente no se vería compensado, con lo que sólo comprobaremos los elementos de la diagonal de la Hessiana.

- **El Método de Newton Intervalar** (Paso 9)

El método de Newton intervalar descrito en la sección anterior se puede utilizar para determinar todos los ceros dentro de la caja X que se este procesando del gradiente de la función objetivo f. Evaluando f sobre todas las cajas solución, se pueden eliminar aquellos puntos estacionarios de f que no sean el mínimo global.

Este procedimiento es muy ineficiente, puesto que lo que no se desea es determinar todos los puntos estacionarios de f. Se utiliza el método de Newton intervalar para realizar gran parte del trabajo a realizar en la resolución del problema de optimización, pero se utilizan las técnicas vistas en los puntos anteriores, en concreto, la utilización del gradiente, la utilización de la cota superior y el test de no convexidad para reducir parte del trabajo. A estos tres técnicas se las conoce como procedimientos de eliminación porque su función es eliminar toda o parte de la caja X que se está procesando.

La primera técnica basada en el gradiente trata de eliminar una caja, si lo consigue se obtiene el mismo resultado que se obtendría utilizando el método de Newton intervalar pero con mucho menos esfuerzo de cálculo. La segunda técnica basada en el test de no convexidad, puede eliminar cajas que no contienen ningún punto estacionario con menos esfuerzo de cálculo que el método de Newton intervalar, y a la vez puede eliminar cajas que contengan puntos estacionarios de f que no sean mínimos, en cambio, el método de Newton no podría. Finalmente, la tercera técnica elimina subcajas en las que f es mayor que la cota superior $\overline{f}$ del mínimo global de f, pudiendo de esta manera eliminar cajas que contienen mínimos locales de f, mientras que el método de Newton intervalar no podría.

Al utilizar las técnicas anteriores en conjunción con el método de Newton intervalar, no se utilizará dicho método hasta que la iteraciones converjan en punto, puesto que dicho punto podría resultar no ser un mínimo global, sino que se utilizará dicho método a un paso para a continuación aplicar otros procedimientos antes de aplicar una nueva iteración del algoritmo. A este método de Newton intervalar simplificado se le conoce como método de Newton intervalar especial.

Básicamente los pasos de dicho método especial son los siguientes:

Algoritmo 7: Algoritmo de Newton Intervalar Especial

Paso 1. Inicialmente partimos de una caja X que es la que se esta procesando. Sobre dicha caja se calcula el punto medio x = m(X), para a continuación determinar J(x,X) a partir de la descomposición serie de Taylor de la función f. Una vez calculado J(x,X) calculamos una aproximación J^c para el centro. Sobre dicha aproximación determinamos su inversa aproximada B utilizando un procedimiento que nos permita calcularla aún en el caso de que J^c sea una matriz singular. Finalmente calculamos M(x,X) = BJ(x,X), f(x) y r(x)=Bf(x).

Paso 2. Si M(x,X) es diagonal dominante, saltamos al paso 5.

Paso 3. Realizamos una iteración del método de Gauss-Seidel para resolver M(x,X)(y-x)=Bf(x) para y. Durante dicha iteración, realizamos la siguiente intersección: X'=X $\cap$ y que dos devuelve la caja X'.

Paso 4. Regresamos al algoritmo principal.

Paso 5. Tratamos de resolver M(x,X)(y-x)=Bf(x) para y utilizando el método de eliminación de Gauss ya sea en forma directa o mediante factorización triangular de M(x,X). Si el proceso falla debido a la división de algún intervalo que contenga el cero, saltamos al paso 3, mientras que si el proceso finaliza con éxito, nos devolverá una caja resultante que denotaremos mediante X' y volveremos al algoritmo principal

- **Acotación del Mínimo** (Paso 13)

A continuación vamos a describir como determinar una cota inferior menor $\underline{f}$ del mínimo global f^*. Supongamos que se ha ido aplicando el algoritmo de optimización de Hansen utilizando los procedimientos descritos hasta el momento, eliminando la mayoría de la caja inicial $X^{(0)}$. Si sólo quedan s subcajas de $X^{(0)}$, todas de ella de tamaño pequeño: $X^{(i)}$ (i=1,...,s), puesto que el mínimo global x^* no puede haber sido borrado, debe estar contenido en alguna de dichas cajas.

Calcularemos $f(X^{(i)})$ para i=1,...,s de forma que $[c_i, d_i] = f(X^{(i)})$. Tal como se ha descrito anteriormente, se dispondrá de una cota superior $\overline{f}$ del mínimo global. Si para algún i = 1,...,s determinamos que $c_i > \overline{f}$, entonces $X^{(i)}$ no puede contener el mínimo global x^*, puesto que

$$f(x) \geq c_i > \overline{f} \geq f^* \qquad (9.25)$$

para $x \in X^{(i)}$. Por lo tanto, se puede eliminar la subcaja $X^{(i)}$. Una vez borradas todas las subcajas de forma que se cumpla que

$$c_i \leq \bar{f} \qquad (9.26)$$

para i=1,...,s. Entonces, definiremos

$$\underline{f} = \min_{1 \leq i \leq s} c_i \qquad (9.27)$$

Entonces

$$\underline{f} \leq f^* \leq \bar{f} \qquad (9.28)$$

sea, se dispone de las cota superior e inferior del mínimo global f^*. Supongamos que

$$\underline{f} - \bar{f} \leq \varepsilon \qquad (9.29)$$

para algún $\varepsilon > 0$. En el apartado que presentaremos a continuación mostraremos como asegura dicha condición. Entonces puesto que

$$\underline{f} \leq f(x) \leq \bar{f} \qquad (9.30)$$

para cualquier punto x en de cualquiera de las cajas que no se han eliminado, tendremos que

$$f(x) - f^* \leq \varepsilon \qquad (9.31)$$

para cualquier x. Por lo tanto, f(x) es una buena aproximación para f^* para cualquier punto de cualquiera de las cajas que no se han eliminado.

- **Finalización del Algoritmo** (Paso 12)

Puesto que el algoritmo de optimización que estamos utilizando subdivide la caja inicial $X^{(0)}$ en subcajas, el número de subcajas almacenadas, y por lo tanto pendientes de procesar, puede crecer. A simple vista puede parecer que dicho número de cajas creciera de forma exponencial, sin embargo, en la práctica esto no sucede. Las cajas normalmente son completamente eliminadas, de forma que el número de cajas que se conservan es bastante reducido.
Se dispone de dos condiciones que se deben cumplir para dar por finalizado el algoritmo de optimización. La primera condición es que

$$w(X) \leq \varepsilon_X \qquad (9.32)$$

para cualquiera de las cajas que no han sido eliminadas por el algoritmo. La segunda condición que se debe cumplir es que

$$\bar{f} - \underline{f} \leq \varepsilon_F \qquad (9.33)$$

Esta última condición asegura que f^* este acotada dentro de la tolerancia ε_F. Se supone implícitamente que dicha tolerancia no se escogerá tan pequeña que los errores de redondeo impidan satisfacer la condición anterior.

- **La Lista de Cajas** (Paso 1 y 10)

El algoritmo normalmente empieza su búsqueda sobre una determinada caja inicial $X^{(0)}$, aunque podría también empezar sobre un conjunto de cajas dentro de las cuales se busca el mínimo global. No existe ninguna desventaja si la región de búsqueda está formada por cajas que no conectadas entre sí. Simplemente colocaremos las cajas iniciales en la lista L_1 de cajas a procesar.

A medida que el algoritmo progresa, generalmente la caja inicial $X^{(0)}$ se divide en subcajas. Dicha subdivisión se puede hacer mediante subdivisión directa, o bien, eliminando agujeros en una componente de una caja formada utilizando el método intervalar de Newton. A medida que se generan

nuevas cajas, se colocan en dos listas. La primera lista L_1 consiste de cajas X para las cuales no se cumple uno o ambos criterios de finalización

$$w(X) \leq \varepsilon_X \qquad w[f(X)] \leq \varepsilon_F \tag{9.34}$$

La segunda lista L_2 consiste de cajas X para las cuales se cumplen ambos criterios de finalización.

- **Elección de la Caja a Procesar** (Paso 1 y 10)

Siempre que el algoritmo de optimización genera una nueva caja X, esta se coloca en la lista de cajas apropiada. Cuando el ciclo del algoritmo principal empieza, la caja a procesar se escoge de la lista L_1. A continuación vamos a describir como se realiza dicha elección.

Antes de que la caja se coloque en la lista L_1, se habrá evaluado f(X). Sea $[f^L(X), f^R(X)]$ el resultado de dicha evaluación. La caja a procesar escogida de la lista L_1 será aquella que tenga el menor $f^L(X)$. Si la caja X es pequeña, este procedimiento tiende a escoger cajas que contengan puntos en los cuales la función f sea igual o se aproxime al mínimo global f^*, aumentado la velocidad de convergencia puesto que el procedimiento explicado en el apartado correspondiente a la utilización de la cota superior del mínimo global para eliminar cajas funciona mejor cuanto menor es. Mientras que si X es grande, entonces $f^L(X)$ tiende a ser mucho menor que el valor de f en X debido al problema de la dependencia entre variables al evaluar f. Por lo tanto, cuando L_1 contiene cajas grandes, el procedimiento tiende a escoger ya sea una caja en al cual f es pequeña o una caja en la cual se deba reducir su tamaño para obtener información más precisa sobre la función f y sus derivadas. La reducción del tamaño de X se producirá ya sea por algoritmo de optimización, o porque, se divide en partes.

- **Subdivisión de una Caja** (Paso 11)

En un paso u otro, el algoritmo de optimización evalúa la función objetivo, su gradiente y su Hessiana sobre la caja que se está procesando. Cuanto menor sea esta caja, menor será las variación de la función, el gradiente y la Hessiana sobre la misma. Además, cuanto menor sea la caja, mejores serán las cotas que se obtendrán al evaluar el rango de dichas funciones sobre dicha caja. Por lo tanto, el algoritmo de optimización es tanto más eficiente cuanto menor será la caja que se está evaluando.

Cuando el algoritmo no consigue reducir significativamente el tamaño de una caja X, esta se debe dividir en partes dividiendo una o más componentes de X por la mitad. Si se dividieran simultáneamente las k componentes, se generarían 2^k subcajas. Si k es grande, se generaría un número excesivo de subcajas. Normalmente se escoge k=3, aunque es factible escoger un valor un poco mayor. Dicha elección creará ocho nuevas cajas. Sin embargo, si n=1 o n=2 entonces se realiza una sola subdivisión en cada dimensión.

Supongamos el siguiente caso. Si aplicamos el algoritmo de optimización sobre la caja inicial y no se obtiene ningún progreso, entonces dividimos la caja en una dimensión y aplicamos el algoritmo a caja subcaja. Si el algoritmo no consigue progresar en ninguna de las subcajas, se vuelve a dividir cada subcaja por la mitad de nuevo. Este proceso se repite hasta que tengamos 2^k cajas y se haya aplicada el algoritmo sobre cada una de ellas. En total, el algoritmo se habrá aplicado $2^{k+1} + 1$ veces. Ahora supongamos que se realizan todas las divisiones sin aplicar el algoritmo de optimización hasta que se hayan generador las 2^k cajas. Entonces, se aplica el algoritmo. En este caso se habrá aplicado el algoritmo 2^k veces. Obsérvese que el primer caso el trabajo es siempre menos de dos veces el trabajo que se debe realizar en el segundo caso.

Por lo tanto, teniendo la cantidad de trabajo necesario, en principio es bastante indiferente la forma de proceder. Sin embargo, donde el algoritmo precisa una gran cantidad de cálculo es cuando aplica el método de Newton intervalar. Si una subcaja puede ser eliminada aplicando cualquiera de los métodos de eliminación de subcajas que hemos presentado, entonces la aplicación del método de Newton no es necesaria evitando una gran cantidad de cálculos. Cuanto menor sea la caja, más probable será que no debamos aplicar el método de Newton intervalar. Por lo tanto, puede resultar interesante el subdividir cada caja en subcajas antes de aplicar el algoritmo. Aunque por otro lado, cada aplicación del algoritmo

reduce el tamaño de la caja sobre la que se aplica, por lo tanto, aplicando el algoritmo cada vez que una caja ha sido subdivida también tiene ventajas.

Al escoger la componente de la caja a subdividir, el algoritmo debe intentar determinar aquella sobre la cual la función f varía más. Sea $x_i = m(X_i)$ para i=1,...,n y definamos

$$F_i(t) = f(x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_n) \qquad (9.35)$$

Como una medida de cuanto varía f cuando x_i varía sobre X_i, se podría utilizar

$$W_i = \max_t F_i(t) - \min_t F_i(t) \qquad (9.36)$$

donde el máximo y el mínimo son sobre $t \in X_i$. Sería razonable pues subdividir la j-ésima componente de X si

$$W_j = \max_i W_i \qquad (9.37)$$

para i=1,...,n. Debe notarse que W_i depende sólo de un único valor de x_k para cuyos índices k=1,...,n con $k \neq i$. En principio, W_i no será el mismo para otros valores en X_k de estas variables x_k. Sin embargo, la determinación de W_i es otro problema de optimización, aunque en la práctica puede ser aproximada por

$$w[F_i(X_i)] \qquad (9.38)$$

Por lo tanto, se subdividirá la caja X_i por el valor de i que tenga $w[F_i(X_i)]$ mayor. El problema es que para determinar la componente de X_i para el cual W_i es mayor, deberíamos calcular los n intervalos $F_i(X_i)$ para i=1,...,n. En su lugar, se propone utilizar otros resultados utilizados para otras finalidades, reduciendo el esfuerzo de cálculo total. A continuación vamos a ver como funciona esta forma alternativa de determinación de la componente de la caja X_i a subdividir.

Sea $X_i=[a,b]$ la caja a subdividir, entonces se define

$$G_i(t) = g_i(x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_n) \qquad (9.39)$$

donde g_i es la i-ésima componente del gradiente de f. Así mismo se define,

$$\begin{aligned} m_i &= \min_{t \in X_i} G_i(t) \\ M_i &= \max_{t \in X_i} G_i(t) \end{aligned} \qquad (9.40)$$

De donde, es obvio que se cumple

$$\begin{aligned} \max_{t \in X_i} F_i(t) &\leq F_i(a_i) + \int_{a_i}^{b_i} M_i du \\ \min_{t \in X_i} F_i(t) &\geq F_i(a_i) + \int_{a_i}^{b_i} M_i du \end{aligned} \qquad (9.41)$$

por lo tanto, puesto que

$$M_i - m_i \leq w[G_i(X_i)] \qquad (9.42)$$

tendremos que

$$\max_{t \in X_i} F_i(t) - \min_{t \in X_i} F_i(t) \leq w[G_i(X_i)] w(X_i) \qquad (9.43)$$

En esta relación, las variables x_j para j=1,...,n y $j \neq i$ aparecen implícitamente como $m(X_j)$. Si se reemplaza cada x_j por X_j en el miembro derecho, la relación anterior es cierta para cualquier $x_j \in X_j$ con j=1,...,n y $j \neq i$ en el miembro izquierdo. O sea, para todo x_j

$$\max_{t\in X_i} F_i(t) - \min_{t\in X_i} F_i(t) \le D_i(X) \qquad (9.44)$$

donde

$$D_i(X) = w[g_i(X)]w(X_i) \qquad (9.45)$$

para i=1,...,n. Las desigualdades utilizadas para derivar la expresión distan bastante de ser precisas. Por lo tanto, no se puede afirmar que el índice i para el cual $D_i(X)$ sea el mayor, sea también el tiene mayor W_i. Sin embargo, se espera que exista una buena correlación entre ambas cantidades. Por lo tanto, en el algoritmo de optimización se supondrá que la componente de X para la cual $D_i(X)$ sea la mayor es la más adecuada para realizar la subdivisión.

Por lo tanto, para determinar cual es la componente a subdividir, se debe calcular $g_i(X)$ para i=1,...,n. Esto no implicará ningún cálculo adicional, puesto que dichos cálculos también se deben realizar para poder aplicar el test de monotonicidad. Esta es la razón fundamental para utilizar este procedimiento para seleccionar la componente a subdividir.

9.7.2 Optimización con restricciones

A continuación vamos a describir como se debe modificar el algoritmo de Hansen para problemas de optimización sin restricciones cuando se desea aplicar sobre problemas de optimización con restricciones. El problema que pretendemos resolver es el siguiente:

$$\begin{aligned} &\text{min global } f(x) \\ &\text{sujeto a } p_i(x) \le 0 \quad \text{para } i = 1,\cdots,m \\ &\qquad\quad q_i(x) \le 0 \quad \text{para } i = 1,\cdots,r \end{aligned} \qquad (9.46)$$

donde se supone que f(x) es dos veces continuamente diferenciable y que las restricciones $p_i(x)$ y $q_i(x)$ son continuamente diferenciables.

Tal como en el caso de optimización sin restricciones, se supone que la caja inicial donde se desea determinar el mínimo global $X^{(0)}$ viene dada, de forma que el algoritmo de optimización determinará el mínimo global de f(x) sobre dicha caja inicial sujeto a las restricciones dadas. Si alguna de las restricciones desigualdad es de tipo cota

$$a - x_i \le 0 \qquad (9.47)$$

o bien,

$$x_i - b \le 0 \qquad (9.48)$$

entonces dichas restricciones determinan alguna de las caras de $X^{(0)}$. Cualquier otra cara de $X^{(0)}$, escogida por el usuario del algoritmo de optimización, no se considera una restricción sino simplemente un límite del espacio de búsqueda del algoritmo.

El enfoque utilizado para resolver este problema de optimización es el mismo que para el caso anterior. El algoritmo va eliminando subcajas de la caja original $X^{(0)}$ que no puedan contener la solución global. El algoritmo se detiene cuando las cotas sobre x^* y f^* son suficientemente estrechas. Ambos algoritmos utilizan las mismas subrutinas pudiéndose utilizar un único programa para resolver ambos tipos de problema.

La diferencia fundamental entre el caso con restricciones del caso sin restricciones, es que un punto para ser un mínimo global o local es que el primer caso se deben de cumplir las condiciones de Fritz-John en dicho punto, o bien, alternativamente las condiciones de Kunh-Tucker.

Las condiciones de Fritz-John se pueden escribir de la siguiente manera:

$$
\begin{aligned}
& u_0 \nabla f(x) + \sum_{i=1}^{m} u_i \nabla p_i(x) + \sum_{i=1}^{r} v_i \nabla q_i(x) = 0 \\
& u_i p_i(x) = 0 \quad \text{para } i = 1, \cdots, m \\
& q_i(x) = 0 \quad \text{para } i = 1, \cdots, r \\
& u_i \geq 0 \quad \text{para } i = 1, \cdots, m
\end{aligned}
\qquad (9.49)
$$

donde $u_0,...,u_m$, $v_1,...,v_r$ son multiplicadores de Lagrange. Las condiciones de Fritz-John normalmente no incluyen una condición de normalización, por lo tanto, habrá más variables que ecuaciones.

Las condiciones de Fritz-John difieren de las condiciones de Kuhn-Tucker, utilizadas más normalmente, en la inclusión del multiplicador de Lagrange u_0. Si se fuerza a que $u_0=1$ y se omiten las condiciones de normalización, lo que queda son exactamente las condiciones de Kuhn-Tucker. Cuando se utilizan estas últimas condiciones se supone que las restricciones no son linealmente dependientes en el mínimo.

A continuación veamos de que forma se pueden utilizar las restricciones dentro del algoritmo de optimización para eliminar subcajas que no puedan contener el mínimo global. Cuando se aplican las condiciones de Fritz-John, el algoritmo estará buscando una solución x^* en una determinada caja X. Si el problema de optimización a resolver tiene una restricción igualdad que no se satisface en ningún punto de la caja X, entonces dicha caja se puede eliminar. Por lo tanto, dada una caja X, evaluaremos la función restricción igualdad q_i para i=1,...,r sobre X, obteniendo el siguiente resultado

$$q_i(X) = [q_i^L(X), q_i^R(X)] \qquad (9.50)$$

Si $q_i^L(X) > 0$, o bien, $q_i^R(X) < 0$ para cualquier i=1,...,r, entonces no existe ningún punto $x \in X$ en el cual $q_i(x)=0$, y por lo tanto se puede eliminar X.

De forma similar, evaluaremos las restricciones desigualdad sobre X, obteniendo

$$p_i(X) = [p_i^L(X), p_i^R(X)] \qquad (9.51)$$

Si $p_i^L(X) > 0$ para algún i=1,...,m, entonces $p_i(x) > 0$ para todo $x \in X$. O sea, no existe ningún punto $x \in X$ en el cual la condición $p_i(x) \leq 0$ sea satisfecha. Por lo tanto, se puede eliminar X. Mientras que si $p_i^R(X) < 0$ para algún i=1,...,m, entonces $p_i(x) < 0$ para todo $x \in X$. Por lo tanto, esta restricción en particular no está activa para ningún punto de X, pudiéndose ignorar cuando se esta buscando una solución dentro de dicha caja.

Por lo tanto, antes de comprobar si se cumplen las condiciones de Fritz-John sobre la caja X que se está procesando, se aplicarán los tests anteriores, de forma que al aplicar las condiciones de Fritz-John sobre la caja sólo se tendrán en cuenta aquellas restricciones desigualdad que estén activas.
Para verificar las condiciones de Fritz-John sobre la caja se utilizará el método de Newton intervalar para determinar si existe algún punto en dicha caja que las verifique.

A continuación se presenta el algoritmo de Hansen para el caso de que se desee aplicar a un problema de optimización con restricciones desigualdad. En la descripción de dicho algoritmo se puede apreciar como se utilizan las restricciones y las condiciones de Fritz-John para eliminar subcajas de la caja original que no pueden contener el mínimo global.

Algoritmo 8: Algoritmo de Hansen con restricciones

Paso 1. Para cada caja X contenida en la lista L_1, se determina su centro aproximado x. Evaluación de la función f en el centro x de cada una de las cajas contenidas en las lista L_1. El resultado de dicha evaluación se utilizará para actualizar el valor de $\bar{f}$.

Paso 2. Para cada caja X contenida en la lista L_1, si $f(x) \leq \bar{f} \leq \infty$ siendo x el centro de las cajas, entonces realizaremos una búsqueda lineal para trata de reducir $\bar{f}$.

Paso 3. Para cada caja inicial X contenida en la lista L_1, evaluaremos f(X). El resultado de dicha evaluación proporcionará $[f^L(X), f^R(X)]$.

Paso 4. Eliminación de cualquier caja X en L_1 para la cual $f^L(X) > \bar{f}$.

Paso 5. Si la lista L_1 está vacía, entonces saltaremos al paso 17. En caso contrario, elegiremos la siguiente caja X de la lista L_1 que será procesada por el algoritmo que tenga el menor valor de $f^L(X)$, y se eliminará de la lista L_1.

Paso 6. Evaluación de las restricciones sobre la caja X: $p_i[X] = [p_i^L(X), p_i^R(X)]$. Si $p_i^L(X) > 0$ para cualquier i = 1, ... , m (es decir, X no es una región factible), saltaremos al paso 5. En caso contrario, para cada i = 1, ... , m, determinaremos el conjunto I para el cual $0 \in p_i(X)$. Si el conjunto I está vacío, entonces X es una región factible con toda seguridad. Si I no está vacío, saltaremos al paso 9.

Paso 7. Test de monotonicidad. Si $0 \notin g_i(X)$ para algún i = 1, ... , n saltaremos al paso 5.

Paso 8. Test de no convexidad. Si $H_{ii}(X) < 0$ para cualquier i = 1, ... , n saltaremos al paso 5.

Paso 9. Si $\bar{f} = \infty$, saltaremos al paso 10. En caso contrario, evaluaremos una aproximación del centro x de la caja X, así como f(x) utilizando la aritmética intervalar. Si $f^R(x) < \bar{f}$, saltaremos al paso 10. En caso contrario, aplicaremos el método cuadrático que utiliza la desigualdad $f(x) \leq \bar{f}$ para borrar toda la caja X, o bien, sólo una parte. Si la caja resultante, a la que denominaremos de nuevo X, está vacía saltaremos al paso 5. En caso contrario, registraremos todas las partes que puedan ser borradas de la caja resultante y saltaremos al paso 10.

Paso 10. Elección de las restricciones que deben ser aplicadas. Adición de la restricción $f(x) - \bar{f} \leq 0$ si se cumple:

$$\frac{f^R(X) - \bar{f}}{f^R(X) - f^L(X)} > 0.25$$

Si no aplica ninguna restricción, saltaremos al paso 11. En caso contrario, se linealizará el conjunto de restricciones resultantes. Resolución del problema de optimización aplicando el conjunto de desigualdades lineales resultantes. Si el conjunto solución está vacío, saltaremos al paso 5. En caso contrario, registraremos todas las partes de la caja solución que puedan ser borradas y saltaremos al paso 11. Si las parte que pueden ser borradas se solapan con las que se han generado en el paso 9 entonces, se mezclarán unas con otras.

Paso 11. Modificación de las condiciones de John incluyendo las funciones restricción p_i cuyos índices pertenezcan al conjunto I obtenido en el paso 6. Aplicación de un paso del método intervalar de Newton a las condiciones de John modificadas. Si el conjunto solución está vacío, entonces saltaremos al paso 5. En caso contrario, saltaremos al paso 12. En caso de que el método intervalar de Newton pruebe la existencia de una solución, esta información debe registrarse.

Paso 12. Si $w(X) < \varepsilon_X$ entonces colocaremos X en una lista temporal L' y saltaremos al paso 13. En caso contrario, mezclaremos las partes de la caja X que se hayan generado en el paso 11 con las parte generadas en pasos anteriores. Si existen partes que satisfagan las condiciones necesarias para ser subdividas, utilizaremos las tres más grandes para dividir la caja X en subcajas, colocándolos en la lista L'. Si no existe ninguna parte de la caja X que pueda utilizarse para subdividir la caja X en subcajas, saltaremos al paso 15.

Paso 13. Actualización de $\bar{f}$ a partir de cada una de las cajas contenidas en la lista temporal L' tal como se describe en los pasos 1 y 2.

Paso 14. Para cada caja X de la lista temporal L', evaluaremos w(X) y f(X). Si $w(X) < \varepsilon_X$ y $w[f(X)] < \varepsilon_F$, entonces colocaremos X en la lista L_2. En caso contrario, colocaremos X en la lista L_1. Saltaremos al paso 5.

Paso 15. Si $w(X) < \varepsilon_X$ y $w[f(X)] < \varepsilon_F$, entonces colocaremos X en la lista L_2. Saltaremos al paso 5. En caso contrario, saltaremos al paso 16.

Paso 16. Si X es una versión suficientemente reducida de la caja con la que se empezó en el paso 5, colocaremos la caja X en la lista L_1 y saltaremos al paso 5. En caso contrario, X en dos para tres de las componentes. Colocaremos las cajas resultantes en la lista L' y saltaremos al paso 13.
Paso 17. Para cada caja X de la lista L_2, habremos evaluado f(X) en el paso 14, obteniendo $[\underline{f}^L(X), \underline{f}^R(X)]$. Borraremos cualquier caja de la lista L_2 que cumpla $\underline{f}^L(X) > \overline{f}$. Sean $X^{(1)}, \ldots, X^{(s)}$ las cajas que no se han eliminado, donde s es el número de dichas cajas. Determinaremos

$$\underline{f} = \min_{1 \le i \le s} f^L(X^{(i)})$$

y

$$\underline{f}^R = \min_{1 \le i \le s} f^R(X^{(i)})$$

Paso 18. Finalización del algoritmo. Si después de la finalización del algoritmo $\overline{f} < \infty$ entonces sabemos que existe un punto factible en la caja inicial. Por lo tanto, podemos afirmar que el mínimo de la función f^* cumple:

$$\underline{f} \le f^* \le \overline{f}$$

y que, para cualquier punto x de caja de una las cajas que no se han eliminado:

$$f(x) - f^* \le \varepsilon_F$$

Además, para cada una de estas cajas se cumplirá también:

$$w(X) \le \varepsilon_X$$

Mientras que si al finalizar el algoritmo $\overline{f} = \infty$, significa que no se ha encontrado ningún punto factible. Sin embargo, podemos afirmar que si existe algún punto factible en $X^{(0)}$, entonces el mínimo absoluto de la función f^* debe cumplir

$$\underline{f} \le f^* \le \overline{f}^R$$

verificándose

$$f(x) - f^* \le \varepsilon_F$$

para cualquier punto x de caja de una las cajas que no se han eliminado.

9.8 Solvers de Optimización Global basados en el Algebra de Intervalos

- **GIA InC++: Global Interval Arithmetic and Optimization Library**

GIA InC++ es una librería de funciones realizada en C++ que permite obtener el rango de valores que toma una función intervalar. Dicha librería se basa en los algoritmos de branch and bound propuestos por Ratscheck y Rokne [Ratscheck 84] [Ratscheck 88] y por Hansen [Hansen92].

Las principales características de dicha librería son:

1. Optimización Algebraica

Utiliza un preprocesador algebraico previo a la aplicación de los algoritmos de optimización que determina la extensión intervalar óptima de la función de la cual se desea evaluar el rango de valores. Este preprocesado previo reduce mucho el tiempo de cálculo incluso si se tiene en cuenta el tiempo necesario para realizar las manipulaciones algebraicas.

2. Aritmética Intervalar Extendida

Los argumentos de las funciones pueden ser cualquiera de los tipos de intervalos definidos en la librería LIA InC++, que se ha presentado en este mismo capítulo como una de las posibles implementaciones

prácticas de la aritmética intervalar. Dicha implementación de la aritmética intervalar permite manejar intervalos abiertos, con extremos infinitos e intervalos no conexos (multiintervalos).

3. Reutilización de los Cálculos

GIA InC++ puede reutilizar los cálculos numéricos anteriores de diversas formas con el objetivo de minimizar los cálculos innecesarios. La implementación de técnicas de reutilización de cálculos permite reducir también el tiempo de cálculo.

4. Control Flexible de los Parámetros de Cálculo

La complejidad de cálculo de las evaluación global de funciones intervalares es en general impredecible. Por ello, GIA InC++ proporciona facilidades en el control de los parámetros de cálculo necesarias en las aplicaciones prácticas Los principales parámetros que se pueden controlar incluyen:

- *Precisión*. GIA acota el rango de valores real de la función dentro de un criterio de precisión absoluta o relativa definido por el usuario. Además, proporciona una aproximación por defecto del resultado. Las cotas inferior y superior conjuntamente indican cual es la precisión de la evaluación que se ha realizado.
- *Tiempo Límite*. El algoritmo de cálculo se puede interrumpir si alcanza un tiempo límite fijado por el usuario. Alcanzado dicho tiempo límite, GIA devuelve las cotas inferior y superior como resultado. Esta posibilidad es bastante interesante puesto que permite establecer un compromiso entre la precisión y el tiempo de cálculo en aplicaciones en sistemas que deban funcionar en tiempo real.
- *Límite de Divisiones*. El algoritmo de branch and bound se puede controlar también acotando el tamaño del espacio de búsqueda.
- *Modo Algebraico*. Dispone de diversos mecanismos de optimización algebraica, permitiendo los requerimientos de tiempo y espacio entre las optimización algebraica (tiempo de definición) y la evaluación numérica (tiempo de evaluación): a mayor optimización algebraica menor será el tiempo de ejecución, y viceversa.

La utilización de la librería GIA InC++ contempla los siguientes pasos: (ver el Apéndice de este capítulo para ver una descripción más detallada de la utilización de esta librería)

Utilización de la librería GIA InC++

1. Construcción del objeto función intervalar, al que denominaremos por ejemplo E, correspondiente a la función $Y = F(X_1, \cdots, X_2)$.
2. Fijar los valores de los argumentos intervalares $X_1, \cdots, X_n$.
3. Modificar opcionalmente los parámetros de cálculo del objeto E si es necesario.
4. Evaluación de E.
5. Lectura del resultado.

9.9 Aplicación a la Generación de Envolventes

La aplicación de la optimización global basada en el álgebra de intervalos que se ha presentado en este capítulo a la resolución del problema de optimización planteado al utilizar el nuevo algoritmo de generación de envolventes que se propone en esta tesis pasa por:

Paso 1. Inicialización. Determinar la expresión de la función objetivo a optimizar para cada uno de los estados del sistema del cual se quieren obtener las envolventes. Esta tarea se puede realizar con cualquier programa de manipulación simbólica como MAPLE.
Paso 2. Formulación del problema de forma que el optimizador global basado en el álgebra de intervalos pueda resolver el problema de optimización.
Paso 3. Solución del problema de optimización asociado a cada una de las variables de estado utilizando el optimizador global basado en el algoritmo de branch and bound y el álgebra intervalos
Paso 4. Lectura de los resultados de la optimización y generación de las envolventes en el instante de tiempo actual.
Paso 5. Generación de los nuevos problemas de optimización global asociados a cada una de las variables de estado, para obtener el valor de las envolventes en el siguiente instante de tiempo, actualizando las cotas de los estados al inicio de la ventana temporal, a partir de los resultados obtenidos en la solución de los problemas.
Paso 6. Salto al paso 3.

Gráficamente, esta secuencia de pasos se puede representar de la siguiente manera:

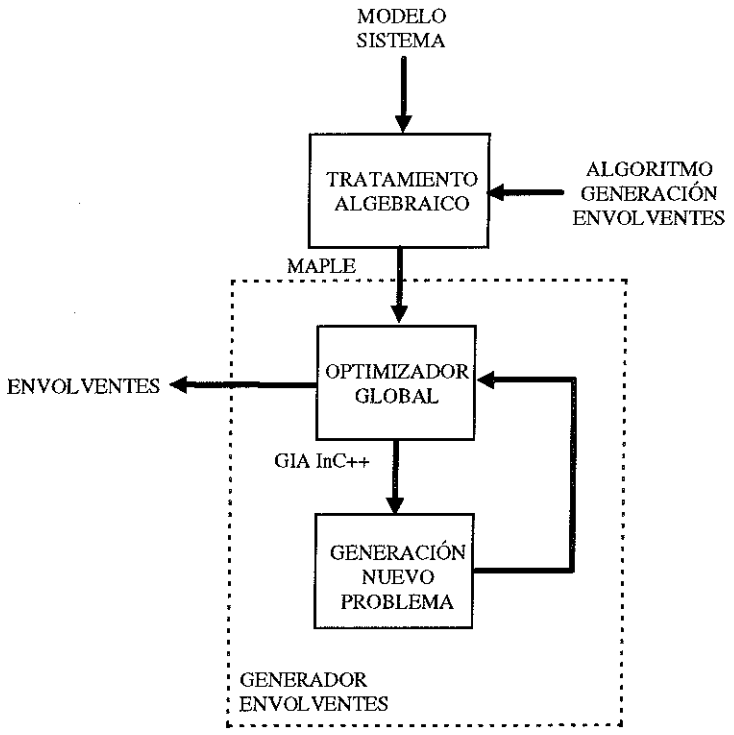


Fig. 9.3 Generación de las envolventes utilizando GIA InC++

En esta tesis se han probado el optimizador global basado en el álgebra de intervalos: GIA InC++. Las características principales del cual se han presentado anteriormente se han presentado anteriormente.

- **Generación de Envolventes utilizando GIA InC++**

Para probar las técnicas de optimización global presentadas en este capítulo para resolver el problema de optimización que aparece al generar las envolventes de un sistema utilizando el nuevo algoritmo de generación de envolventes presentado en esta tesis utilizaremos un sistema de segundo orden cuyo modelo en el espacio de estado discreto es el que se presenta a continuación

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

con la siguiente incertidumbre asociada a los parámetros del modelo:

$$a_1 \in [1.3938, 1.4338]$$

$$a_2 \in [0.5865, 0.6265]$$

Utilizando el algoritmo de generación de envolventes presentado en esta tesis, sobre este ejemplo con una vetaba de longitud L = 5, los problema de optimización global que deben resolverse son los siguientes:

Problema 1: "Envolventes para la variable de estado x_1"

Función Objetivo:

$$x_{1k} = f_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^5 - 4x_{1(k-L)}a_1^3 a_2 + 3x_{1(k-L)}a_1 a_2^2 - a_2 x_{2(k-L)}a_1^4 + 3a_2^2 x_{2(k-L)}a_1^2 - a_2^3 x_{2(k-L)} + a_1^4 - 3a_1^2 a_2 + a_2^2 + a_1^3 - 2a_1 a_2 + a_1^2 - a_2 + a_1 + 1$$

Restricciones:

$x_{1(k-L)} = [\underline{x}_{1(k-L)}, \overline{x}_{1(k-L)}]$

$x_{2(k-L)} = [\underline{x}_{2(k-L)}, \overline{x}_{2(k-L)}]$

$a_1 = [1.3938, 1.4338]$

$a_2 = [0.5865, 0,6265]$.

Problema 2: "Envolventes para la variable de estado x_2"

Función Objetivo:

$$x_{2k} = f_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2) = x_{1(k-L)}a_1^4 - 3x_{1(k-L)}a_1^2a_2 + x_{1(k-L)}a_2^2 - a_1^3a_2x_{2(k-L)} + 2a_1a_2^2x_{2(k-L)} + a_1^3 - 2a_1a_2 + a_1^2 - a_2 + a_1 + 1$$

Restricciones:

$x_{1(k-L)} = [\underline{x}_{1(k-L)}, \overline{x}_{1(k-L)}]$

$x_{2(k-L)} = [\underline{x}_{2(k-L)}, \overline{x}_{2(k-L)}]$

$a_1 = [1.3938, 1.4338]$

$a_2 = [0.5865, 0,6265]$.

In Fig. 9.4 y Fig. 9.5, se representan gráficamente las dos funciones objetivo $x_{1k} = f_1(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2)$ y $x_{1k} = f_2(x_{1(k-L)}, x_{2(k-L)}, a_1, a_2)$, suponiendo que las condiciones iniciales en los estados al inicio de la ventana son igual a cero.

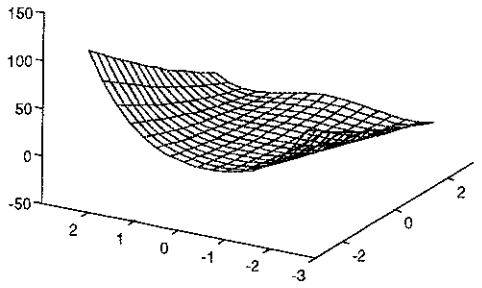


Fig. 9.4 Representación gráfica de la función objetivo f_1 para una ventana de longitud L=5

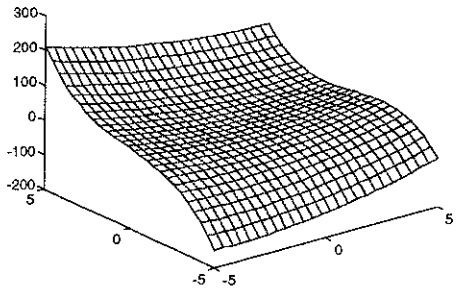


Fig. 9.5 Representación gráfica de la función objetivo f_2 para una ventana de longitud L=5

Para resolver estos dos problemas de optimización global utilizaremos el solver GIA InC++ que basado el algoritmo de búsqueda branch and bound y el álgebra de intervalos, cuyas principales características ya se han presentado en la sección anterior.

Los resultados obtenidos utilizando este optimizador para resolver los problemas de optimización asociados a la generación de envolventes utilizando el algoritmo de generación de envolventes

introducido en esta tesis, utilizando el ejemplo del sistema de segundo orden presentado en esta sección para diferentes longitudes de ventana y empezando siempre en el instante de tiempo k = 0, se muestran la Tabla 1. Las envolventes presentadas en dicha tabla corresponden a la variable de estado x_1. De forma análoga se podrían obtener las envolventes para la variable de estado x_2.

L	Envolvente para el estado x_1	Tiempo de Cálculo
5	[5.18649,6.12189]	0 seg
10	[4.27974,7.05731]	0.33 seg
15	[4.22014,6.51874]	53.72 seg
20	[4.33664,6.51730]	2800.65 seg

Tabla 1 Envolventes para la variable de estado x_1

A partir de estos resultados, se observa que para una ventana de longitud mayor que L = 15, el tiempo de cálculo se incrementa mucho siendo inútil para su aplicación en aplicaciones reales, donde se precisan tiempos de cálculo del orden diez veces inferiores al tiempo de respuesta del sistema que se desee monitorizar. Así por ejemplo, para una longitud de ventana L = 20 el tiempo de cálculo necesario para calcular la envolvente para un estado es de 2800 seg, más de una hora y media. Estos resultados han sido obtenidos con el solver GIA InC++ con una precisión de 10^{-6}, suficiente para considerar que los resultados obtenidos son los exactos. Por supuesto que si se hubieran obtenido unos resultados con una precisión menor, el tiempo de cálculo hubiera sido menor.

A partir de los resultados obtenidos en el capítulo 3, se puede determinar analíticamente para este ejemplo la longitud de ventana mínima estable y la longitud de ventana casi estacionaria. La longitud de ventana mínima estable es la longitud de ventana necesaria para evitar el crecimiento incontrolado de la incertidumbre añadida por el algoritmo de generación de envolventes, mientras que la segunda longitud de ventana es la longitud necesaria para producir aproximadamente, con sólo un incremento del 5% en la incertidumbre total del sistema, los mismos resultados que se obtendrían empezando los cálculos desde el instante inicial k = 0. Para el ejemplo del sistema de segundo orden presentado en esta sección dichas longitudes calculadas analíticamente a partir de las fórmulas del capítulo 3 son:

Mínima Estable	$L_\infty=7$
Casi Estacionaria	$L_{0.05}=15$

Tabla 2. Longitudes de ventana teóricas

A la vista de estos resultados una longitud de ventana L = 15 produce unas envolventes con un incremento del 5% de la incertidumbre total presenten en el sistema. En la Fig. 9.6 se presenta la envolventes correspondiente a la variable de estado x_1 generada utilizando el nuevo algoritmo de generación de envolventes junto con el algoritmo de optimización global de GIA InC++ y utilizando una longitud de ventana L = 15.

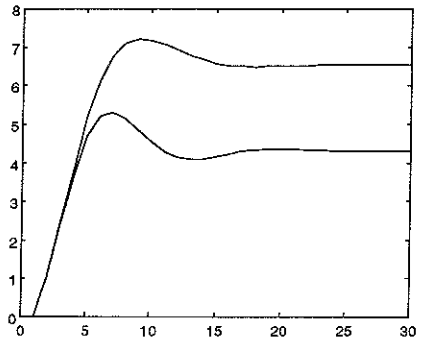


Fig. 9.6 Envolvente para la variables de estado x_1 generada a partir del nuevo algoritmo de generación de envolventes y el algoritmo de optimización global de GIA InC++ con una longitud de ventana L=15

9.10 Conclusiones

En este capítulo se ha presentado un algoritmo de optimización global basado nuevamente en el algoritmo de búsqueda de tipo branch and bound, que utiliza como herramientas de acotación y verificación de la existencia de soluciones en cada una de las particiones del espacio de soluciones a descartar el álgebra de intervalos clásica. Este tipo de algoritmos han aparecido con mucha fuerza en estos últimos años a raiz de los trabajos de Moore, Hansen y Ratscheck/Rokne. Los resultados obtenidos de la aplicación de dichos algoritmos a problemas de optimización global son muy esperanzadores obteniéndose unos tiempos de cálculo mucho mejores que los obtenidos con el resto de técnicas de optimización global. Además, la posibilidad de que este tipo de algoritmos se paralelice, posibilidad, presenta por Leclerc hace que estos tiempos de cálculo aún se puedan reducir más.

Los resultados obtenidos en la aplicación de esta técnica a nuestro problema de optimización global asociado con el problema de generación de envolventes utilizando el nuevo algoritmo de generación presentado en esta tesis también han sido los mejores si los comparamos con el resto de técnicas que se han utilizado, tanto a nivel de precisión como a nivel de tiempo de cálculo.

Esto nos hace tal como se verá en el capítulo final de conclusiones proponer esta técnica como la técnica a utilizar par resolver el problema de optimización global asociado con el problema de generación de envolventes.

Capítulo 10

Capítulo 10:

GENERACIÓN DE ENVOLVENTES COMO UN PROBLEMA DE PROPAGACIÓN DE RESTRICCIONES INTERVALARES

Mientras que las técnicas que hemos visto hasta ahora utilizan el conjunto de restricciones como un todo, por lo tanto, generan soluciones globales, las técnicas que se presentan en este capítulo basadas en la propagación o consistencia de restricciones se basan en la generación de soluciones parciales, que se propagan a todo el conjunto de restricciones.

Se puede decir que las ventajas, que en términos de exactitud y validez proporcionan los métodos de obtención de soluciones globales, se mitigan en la práctica por el hecho que:

(a) Se requieren representaciones rígidas que son difíciles de manipular por no expertos.
(b) Son métodos no generales, ya que sirven para casos especiales e incluso muy especiales, sujetos a condiciones estrictas.
(c) Presentan numerosos inconvenientes en la resolución de problemas complejos.
(d) Frecuentemente están dirigidos por técnicas complejas donde el usuario no tiene control sobre los mecanismos de decisión.

En general, podemos decir que los métodos basados en redes de restricciones son muy útiles para describir aquellos problemas cuya solución requiere satisfacer varias restricciones simultáneas.

10.0 Introducción a los Problemas de Satisfacción de Restricciones (CSP)

- **Redes de Restricciones**

Podemos decir que una **red de restricciones** se define de la siguiente forma

$$R \equiv \begin{cases} C = \{c_1, \cdots, c_s\} \\ V = \{x_1, \cdots, x_n\} \\ D = \{D_1, \cdots, D_n\} \end{cases} \quad (10.1)$$

donde V es el conjunto de las variables de la red, tomando cada $x_i \in V$ valores en un dominio D_i y C es el conjunto de restricciones de la red, y en cada restricción $c_i \in C$ participan un conjunto de variables que denotamos por *Var(c_i)* y su cardinal indica la **aridad de la restricción**.

Cada restricción es una relación definida en un subconjunto de V. Las tuplas de esta relación son todos aquellos valores de las variables que permiten que dicha relación se cumpla. Formalmente, una restricción c_i tiene dos partes:

(a) El **conjunto de variables** de la restricción, *Var(c_i)*
(b) Una **relación** *rel_i* definida sobre *Var(c_i)* que es un subconjunto del producto cartesiano de los dominios de cada una de las variables.

Cada tupla de valores $(v_1, \cdots, v_k)$ de las variables que participan en la restricción c_i y la verifican se denomina una **solución**. Por lo tanto, podemos también caracterizar una restricción c_i por el conjunto de soluciones que la verifican y un conjunto de variables.

Una asignación de un valor único del dominio a cada variable perteneciente a *Var(c_i)* se denomina **instanciación**. Una instanciación es una solución sólo si satisface dicha restricción. El conjunto de

soluciones lo denotamos por *Sol(c_i)*. Cualquier forma de definir conjuntos será adecuada para definir una restricción. Una forma posible consiste en definir el conjunto por extensión, enumerando el conjunto de las tuplas que están en la restricción. Esta forma es adecuada cuando las variables toman valores en dominios discretos, pero no cuando toman valores en dominios continuos. No obstante, se puede expresar este conjunto de valores de forma intensional mediante intervalos, unión de ellos o mediante predicados.

Estas definiciones pueden extenderse al conjunto de restricciones C de la red de restricciones, de tal forma que tenemos

$$\begin{aligned} Sol(C) &= \cap_i Sol(c_i) \\ Var(C) &= \cup_i Var(c_i) \end{aligned} \qquad (10.2)$$

para $i \in \{1,\cdots,s\}$.

Sea un conjunto de variables $x = \{x_1,\cdots,x_n\}$ que toman valores en un dominio D, y un conjunto de valores S, decimos que la **proyección** de S sobre la variable x_i, y lo denotamos por $\Pi_{x_i} S$, es el conjunto de valores

$$\Pi_{x_i} S = \{v_i \mid \exists v_1 \in D_1,\cdots,\exists v_{i-1} \in D_{i-1},\exists v_{i+1} \in D_{i+1},\cdots,v_k \in D_k \text{ para } v \in S\} \qquad (10.3)$$

siendo $v = (v_1,\cdots,v_k)$.

Esto se podría extender a la proyección sobre un conjunto de variables X y entonces la definición anterior de proyección se transforma en

$$\Pi_X S = \Pi_{x_1} S \times \cdots \times \Pi_{x_k} S \qquad (10.4)$$

con $X = (x_1,\cdots,x_k)$.

Dado que la solución de una restricción es un conjunto de valores para las variables que la componen, podemos definir también la **proyección de una restricción** de la siguiente manera:

Sea c una restricción sobre las variables $\{x_1,\cdots,x_n\}$. La proyección de la restricción c sobre la i-ésima variable, que se denota como <c,i>, tal que $(1 \le i \le n)$, es otra restricción sobre la variable x_i y cuyo conjunto solución es

$$Sol(<c,i>) = \{v_i \mid \exists v_1 \in D_1,\cdots,\exists v_{i-1} \in D_{i-1},\exists v_{i+1} \in D_{i+1},\cdots,v_n \in D_n \text{ para } <v_1,\cdots,v_n> \in Sol(c)\}$$

La proyección de una restricción también se representa como $\Pi_{x_i} c$, así se puede decir

$$<c,i> = \Pi_{x_i} c$$

- **Representación de Redes de Restricciones mediante Grafos**

Las propiedades gráficas de las redes de restricciones fueron inicialmente estudiadas para redes binarias. Una red de restricciones binarias es aquella donde cada restricción contiene dos variables. En este caso a la red se le puede asociar un grafo, donde cada nodo representa una variable, y los arcos que lo conectar son aquéllos correspondientes a las variables restringidas. También se usan los hipografos, donde otra vez los nodos son las variables y los hiperarcos, dibujados como regiones, agrupan a las variables que pertenecen a la misma restricción.

Por ejemplo, la representación de una res de restricciones binarias donde A, B, C, D, E representan variables y las restricciones se establecen entre tuplas

$$\{\{A,B\}, [A,C\}, \{D,B\}, \{C,D\}, \{E,B\}\}$$

se representa mediante el grafo de la Fig. 10.1

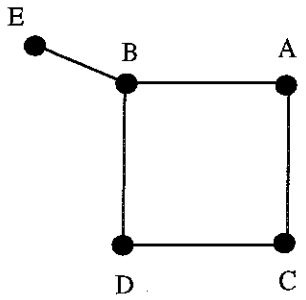


Fig. 10.1. Representación gráfica de una red de restricciones binarias

En general, para restricciones entre más de dos variables como la que se establece entre las variables A,B,C,D

$$A + B \geq C$$
$$A - D \leq B$$
$$A - 2D = C$$
$$D * C < B$$

se puede representar bien como grafos denominados primales, como grafos denominados duales o como grafos generales. En los grafos primales los nodos de nuevo representan variables y se establecen arcos entre las variables pertenecientes a cada restricción, tal como se observa en la Fig. 10.2.

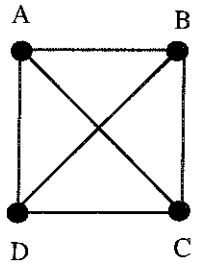


Fig. 10.2. Representación mediante grafos primales

Los grafos duales pueden considerarse como una transformación de redes no binarias en un tipo de red binaria, donde cada restricción representa un nodo, también llamado c-variable y se unen estos nodos con un arco etiquetado con las variables que se comparten entre ellos, tal como se observa en la Fig. 10.3. Por tanto estas redes pueden resolverse como redes binarias.

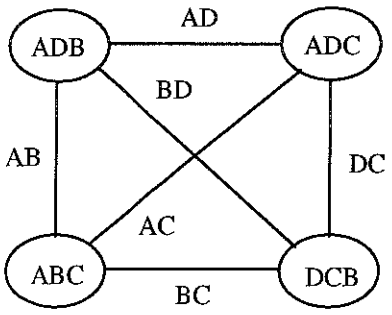


Fig. 10.3. Representación mediante grafos duales

En este trabajo se utilizará una representación para las restricciones en dominios continuos mediante grafos bipartitos con dos tipos de nodos, unos que representan conjuntos de restricciones básicas (nodos rectangulares) y otros que representan los conjuntos de variables (nodos circulares). Los arcos conectan los nodos circulares con los nodos rectangulares que representan las restricciones básicas en la que participan, tal como se muestra en la Fig. 10.4.

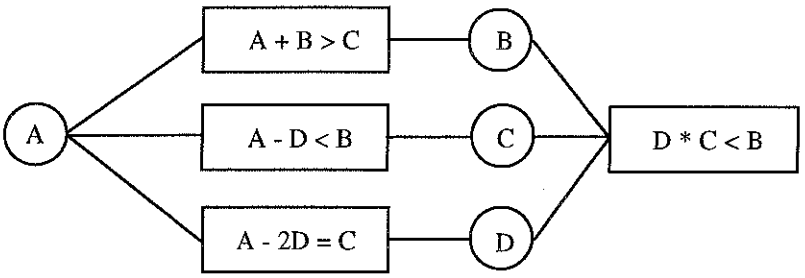


Fig. 10.4. Representación mediante grafos bipartitos

- **Técnicas de Consistencia generales**

La técnica de búsqueda con vuelta atrás, con sus numerosas variantes y mejoras propuesta por Dechter [Dechter90] y algoritmos híbridos realizados con diferentes tipos de las anteriores variantes propuestos por Prosser[Prosser93], aparecen como las principales técnicas para resolver un problema de satisfacción de restricciones (CSP). Pero con objeto de disminuir el espacio de búsqueda se diseñaron un conjunto de algoritmos basados en técnicas de consistencia, de los que uno de los primeros fue el algoritmo AC-3 propuesto por Mackworth[Mackworth77], que se aplica a una red de restricciones para cada uno de los dominios finitos de las variables de las restricciones. En general este algoritmo y sus derivados se denominan **algoritmos de propagación**.

Los algoritmos de propagación obtienen una descripción del conjunto de posibles soluciones de una manera eficiente pero propagando localmente los efectos de las restricciones. Su estructura general es la siguiente:

Algoritmo de Propagación de Restricciones

A := C
P := ∅
Mientras A ≠ ∅
  Sacar la restricción c_1 de A
  D' := ***Refinar***(c_1,D)
  Si D' es vacío
    Salir por no tener solución
  sino si D' ≠ D
    V := *Var*(c_1)
    Para cada restricción $c_i \in P$
      $V' := V \cap Var(c_i)$
      Si $\Pi_{V'}D \neq \Pi_{V'}D'$
        Sacar c_i de P y añadir a A
      Fin Si
    Fin Para
    D := D'
  Fin Si
  Añadir c_1 a P
Fin Mientras

Este algoritmo trata de reducir los dominios de las variables, eliminando aquellos valores que son inconsistentes con las correspondientes restricciones. Se ha demostrado que para dominios finitos y discretos este tipo de algoritmo es de complejidad polinomial. El algoritmo a partir de los dominios de las variables D y teniendo en cuenta las restricciones C obtiene unos nuevos dominios para las variables P. La eliminación de los valores se lleva a cabo mediante el operador ***Refinar***. El algoritmo finaliza cuando encuentra el punto fijo, o bien, cuando no existe solución. El operador ***Refinar(c,D)***, se aplica iterativamente para que todos los valores de los dominios de cada una de las restricciones sean consistentes, en definitiva se consigue que el punto fijo del operador ***Refinar*** para cada restricción en los dominios de las variables.

El operador ***Refinar*** está muy ligado a otro operador, el operador ***Aproximación***, que se define de la siguiente manera:
"Sea un conjunto cualquiera M, y dos subconjuntos μ y μ' de M, el operador de ***Aproximación***, denotado como ***Aprox***, es un operador que obtiene una aproximación del conjunto M y tiene las siguientes propiedades:

(a) *Aproximación del conjunto vacío*: $\boldsymbol{Aprox}(\varnothing) = \varnothing$
(b) *Inclusión*: $\mu \subset \boldsymbol{Aprox}(\mu)$
(c) *Inclusión monótona:* $\mu \subset \mu' \Rightarrow \boldsymbol{Aprox}(\mu) \subset \boldsymbol{Aprox}(\mu')$
(d) *Idempotencia:* $\boldsymbol{Aprox}(\boldsymbol{Aprox}(\mu)) = \boldsymbol{Aprox}(\mu)$

El operador ***Refinar*** para una restricción *c* sobre un dominio D se define como

$$\boldsymbol{Refinar}(c,D) = \boldsymbol{Aprox}\ (Sol(c) \cap D)$$

La idea que subyace en la utilización de este operador consiste en tener una restricción y un dominio para las variables de dicha restricción, y eliminar de manera eficiente los valores de las variables que no satisfacen dicha restricción. La implementación del operador ***Refinar*** será posible, por lo tanto, si una para una restricción *c* encontramos un algoritmo que proyecte el conjunto resultante sobre cada una de las variables. Este operador ***Refinar*** permite, como se expuso en el algoritmo de propagación, propagar por toda la red de restricciones los valores que satisfacen las restricciones. Las propiedades del operador ***Refinar*** son las mismas que las que posee el operador ***Aproximación*** y se derivan de ellas tal como demostró Benhamou [Benhamou97]. De forma resumida puede decirse que el operador es:

(a) *Contractivo*: los elementos refinados son más pequeños que los elementos iniciales.
(b) *Correcto*: cada solución válida de una restricción está dentro de los elementos refinados.
(c) *Monótono*: el operador ***Refinar*** mantiene la propiedad de inclusión al refinar conjuntos incluidos unos en otros.
(d) *Idempotente*: el operador aplicado sobre un elemento ya refinado, produce el mismo resultado.

- **Propiedades del algoritmo de propagación de restricciones**

Antes de establecer las propiedades del algoritmo general de propagación de restricciones que acabamos de presentar, se introducen las siguientes definiciones:

(a) *Red sin solución*. Una red de restricciones R = (C,V,D) se dice que no tiene solución cuando su conjunto de soluciones es vacío, es decir, $Sol(R)=\varnothing$, o bien, cuando existe solución pero no se encuentra dentro del dominio inicial de la variables del problema, es decir, $Sol(R)\cap D=\varnothing$.

(b) *Consistencia local*. Una red de restricciones R = (C,V,D) se dice **consistente localmente** cuando cumple que

$$\forall c_i \in C \quad \text{implica} \quad D = \boldsymbol{Refinar}(c_i, D)$$

Es decir, el nuevo dominio $D' \subseteq D$ obtenido en sucesivos refinados es el mismo. En muchos casos se suele llamar red consistente localmente, red estable o quiescente. La consistencia local siempre supone que el dominio de valores encontrado para las variables contiene la intersección del conjunto solución y el dominio de partida.

(c) *Consistencia global.* Sea una red R = (C,V,D) consistente localmente en un dominio D, podemos decir que dicha red es **consistente globalmente** si no existe otro dominio de valores más pequeño, que también haga la red localmente consistente.

Una vez introducidas estas definiciones podremos decir que el algoritmo de propagación de restricciones tiene las siguientes propiedades:

(a) El algoritmo finaliza, puesto que existe una función de cota asociada al bucle que en cada paso de iteración tiende a cero. Dicha función se basa en el volumen de los hipercubos, que entre una iteración y la siguiente va disminuyendo o se mantiene igual de acuerdo con la propiedad contractiva del operador ***Refinar***. Si se mantiene el volumen del hipercubo inalterable, el cardinal del conjunto de restricciones activas A disminuye, y por lo tanto nos encontramos más cerca de finalizar, ya que la condición de continuación de la instrucción iterativa requiere que dicho cardinal sea distinto de cero.

(b) El algoritmo consigue la consistencia local con carácter general pero no la global. El algoritmo a través del operador ***Refinar*** va instanciando las variables con dominios cada vez más pequeños, que en sucesivas iteraciones harán que dichas variaciones sean inferiores a un valor muy pequeño ε determinado previamente y que hará la red quiescente. Pero no podemos asegurar que dicha consistencia sea global, ya que la aplicación del operador ***Refinar*** se aplica a las restricciones de forma aislada y no en toda la red.

(c) El algoritmo consigue la consistencia global cuando el grafo asociado a la res sea acíclico, o cuando las variables que forman parte de los ciclos tengan valores únicos.

- **Características de los sistemas de propagación de restricciones**

Todos los sistemas de propagación de restricciones comparten el siguiente conjunto de propiedades importantes:

(a) La propagación de restricciones consiste en una simple estructura de control aplicada a un módulo de actualización simple. Por lo tanto, es fácil de codificar, analizar y extender. Es fácil entender sus acciones, explicarlas al usuario y monitorizarlas mediante algún módulo externo inteligente.
(b) Se degrada de forma correcta bajo limitaciones temporales: interrumpiendo el proceso antes de su finalización proporciona información útil disponible hasta ese momento.
(c) Es fácilmente implementable en paralelo, puesto que la actualización puede ser efectuada a través de toda la red de forma simultánea. Excepto para el caso de los sistemas de inferencia de restricciones, es posible implementar cada restricción en un único procesador, que actualiza sus argumentos siempre que cambien, o bien, a intervalos regulares.
(d) Se comporta bien en sistemas incrementales. Se puede añadir una restricción de forma incremental incorporándola a la red, actualizando sus argumentos y propagando sus efectos.
(e) El concepto de red de restricciones como un conjunto de conexiones locales funciona bien con las mismas suposiciones de localidad hechas en los sistemas de inteligencia artificial. Por ejemplo, los programas de razonamiento físico suponen que los efectos físicos se propagan a través de las conexiones entre componentes. Por lo tanto, las restricciones sobre valores físicos relacionarán sólo los parámetros de unos pocos componentes interconectados y cercanos.

- **Clasificación de los sistemas de propagación de restricciones**

Se pueden distinguir seis categorías diferentes de propagación de restricciones según el tipo de información que se actualiza:

- "Constraint Inference", donde se infieren nuevas restricciones y se añaden a la red de restricciones. Ejemplos de dichos sistemas son: ENV de Kuipers [Kuipers84], Quantity Lattice de Simmons [Simmons86] y CMS de Brooks [Brooks81].
- "Label Inference", donde cada nodo se etiqueta con un conjunto de posibles valores. En la asimilación, las restricciones se utilizan para restringir dichos conjuntos.

- "Value Inference", donde los nodos se etiquetan con valores constantes, y las restricciones se utilizan para encontrar valores de los nodos no etiquetados a partir de los nodos etiquetados. Ejemplos de dichos sistemas son: SKETCHPAD [Sutherland63] y THINGLAB [Borning77].
- "Expression Inference" es una generalización de "Value Inference", en el cual los nodos pueden ser etiquetados con un valor expresado como términos de los valores de otros nodos. Cuando un nodo es asignado a dos etiquetas diferentes, estas se igualan y la ecuación resultante debe ser resuelta. Un ejemplo de un sistema de dicho tipo es CONSTRAINTS [Sussman80].
- "Relaxation", donde se asigna a todos los nodos valores exactos, que puede que no sean consistentes con las restricciones. El proceso de asimilación propaga dicho valores a través de la red, hasta que se consigue satisfacer cada una de las restricciones. "Relaxation" ha sido utilizado ampliamente en disciplinas que no son propias de las inteligencia artificial para resolver problemas como el de las ecuaciones en derivadas parciales [Southwell40].
- "Relaxation Labelling", donde se asignan probabilidades de tomar varios valores a cada uno de los nodos. La actualización implica cambios en las probabilidades combinando las anteriores probabilidades de dicho nodo con las probabilidades asignadas a los otros nodos. [Hummel83]

La propagación de restricciones se puede aplicar de forma absoluta o incremental. En la forma absoluta, todas las restricciones están disponibles al principio del procesado, y están fijadas a partir de ese momento. En la forma incremental, se puede añadir nuevas restricciones junto con la respuesta a consultas.

Cada una de las seis técnicas de propagación de restricciones presentan limitaciones y debilidades particulares. "Relaxation" y "Relaxation Labelling" son técnicas deductivas que no son consistentes: las etiquetas derivadas no están lógicamente relacionadas con las anteriores. Por lo tanto, son técnicas difíciles de analizar y de enlazar con sistemas de inferencia más generales. "Value Inference" y "Expression Inference" sólo se pueden utilizar si las restricciones son ecuaciones: no pueden ser utilizadas con desigualdades. "Constraint Inference" y "Label Inference" son difíciles de controlar: puede ser difícil evitar que lleguen a bucles infinitos. En "Constraint Inference", puede ser difícil asegurar que las nuevas restricciones obtenidas sean útiles para la respuesta a consultas. "Label Inference" presenta el atractivo de que se trata de una técnica de deducción consistente, se puede utilizar con restricciones de forma arbitraria y es mucho más fácil de controlo que la inferencia de restricciones general.

10.1 Satisfacción de Restricciones en Dominios Continuos

- **Satisfacción de restricciones con valores numéricos exactos**

Los problemas de satisfacción de restricciones numéricas normalmente funcionan con datos numéricos exactos. Este tipo de sistemas se utilizan para resolver problemas expresados en los siguientes términos:

"Dado un conjunto de ecuaciones E que relaciona un conjunto de variables, donde los valores de las variables de entrada vienen dadas, el problema consiste en determinar el valor de las variables de salida de forma que se satisfagan el conjunto de ecuaciones E".

Por ejemplo, la fórmula de conversión de grados Celsius (C) a grados Fahrenheit (F)

$$F = C * 1.8 + 32$$

se puede representar como una red de restricciones. Si la variable C de entrada, temperatura en grados Celsius, es conocida, entonces la variable F de salida, temperatura en grados Faherenheit se puede calcular mediante dos cálculos locales:

$$X = C * 1.8$$

$$F = X + 32$$

Aplicando, las funciones locales inversas,

$$C = X / 1.8$$

$$X = F - 32$$

el valor de la temperatura en grados centígrados C, se puede calcular a partir de valor de la temperatura en grados Fahrenheit. Mediante esta **propagación de valores locales** cualquier variable de la red de restricciones se puede resolver globalmente a partir de las otras sin tener que resolver las correspondientes ecuaciones algebraicas, lo cual en general es difícil y en el caso general imposible.

Los sistemas de restricciones con valores exactos utilizan también esquemas de relajación iterativa en la resolución del problema, especialmente el **método de Newton**. En este método algunas de las variables de entrada se consideran como variables semilla con valores asignados por el usuario, a partir de los cuales el dicho algoritmo determinará los valores reales a partir de determinar los valores que hacen cero una función f(...) = 0. Por ejemplo, en el caso de una sola variable f(x), un nuevo valor de la solución x_{n+1} para x se puede estimar de forma iterativa mediante

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (10.5)$$

hasta que una solución x_{n+1} que satisfaga que $f(x_{n+1}) \approx 0$ sea alcanzada partiendo de un valor inicial de semilla x_0.

Los sistemas de restricciones con valores exactos tienen las siguientes limitaciones:

(a) *Manejo de datos inexactos.* Este tipo de sistemas aceptan sólo valores exactos como entradas proporcionando valores de salida también exactos. En muchas aplicaciones, sin embargo, los datos de entrada pueden ser ruidosos, inciertos e incompletos. En tales casos, las entradas serán inexactas proporcionando salidas también inexactas, siendo por lo tanto este tipo de sistemas inadecuados para manejar dichas situaciones de inexactitud.
(b) *Resolución de problemas subrestringidos.* La propagación de los valores en un sistema de restricciones con valores exactos puede continuar sólo si existe al menos una restricción local en la cual todas excepto una de las variables relacionadas tiene valores, es decir, la situación no es localmente subrestringida.
(c) *Resolución de problemas sobrerestringidos.* Un problema de satisfacción de restricciones numéricas puede ser también sobrerestringido, es decir, que no tenga soluciones. Sin embargo, normalmente existen muchas posibilidades de hacer el problema resoluble de nuevo. Existen diversas estrategias para seleccionar las variables a modificar.
(d) *Tipos de variables fijas.* En la propagación de valores, el tipo de la variables, es decir, si es de entrada o de salida debe ser fijado antes de realizar el cálculo. Sin embargo, a menudo es imposible realizar dicha distinción. Si un valor dado de la variable es modificado por el motor de propagación, entonces la variable es a la vez una variable de entrada y de salida.

Todos estas limitaciones pueden ser superadas generalizando el sistema de restricciones con valores exactos a un sistemas con valores en intervalos.

- **Satisfacción de restricciones con valores intervalares**

Este tipo de problemas se puede formular en los siguientes términos:

"Dado un conjunto de ecuaciones E que relaciona un conjunto de variables asociadas a dominios intervalares, el problema consiste en refinar dichos dominios tanto como sea posible sin perder soluciones exactas de E, es decir, determinando para cada variable el mínimo subintervalo consistente con su dominio".

La formulación de un problema de satisfacción de restricciones numéricas exactas se puede obtener como un caso particular del problema de propagación de restricciones intervalares, simplemente distinguiendo entre variables de entrada y salida y asignando el dominio intervalar [x,x] a las variables de entrada y el dominio [-∞, +∞] de las variables de salida.

La formulación de un problema de satisfacción de restricciones intervalares difiere de los problemas de optimización matemática clásicos debido a su naturaleza simétrica: el problema consiste en determinar los rangos de todas las variables de un conjunto de ecuaciones de forma simultánea. Además, la búsqueda se realiza a través de intervalos en lugar de a través de puntos exactos.

- **Técnicas de consistencia en dominios continuos**

El objetivo de las diferentes metodologías para obtener soluciones de este tipo de redes de restricciones, consiste en aproximar el conjunto solución de la red de restricciones. La aproximación buscada es

$$\boldsymbol{Aprox}(\mathrm{Sol}(C)\cap I)$$

Si este conjunto es vacío, no existe solución para tal problema. Para implementar el algoritmo de propagación presentado en el apartado anterior para el caso de dominios continuos, se debe primeramente definir los operadores ***Refinar*** más adecuados sobre cada tipo de restricción, así como el operador ***Aproximación*** asociado. Tal como hemos dicho la representación más utilizada para expresar el dominio de las variables reales es un intervalo real.

Cuando se utiliza el algoritmo de propagación definido anteriormente sobre dominios continuos (o sea, intervalos), los resultados que se obtienen son algunas veces demasiado pobres, debido a que las restricciones sobre las que se puede definir el operador de ***Refinar*** con garantías de obtener soluciones correctas son muy pocas. Para restricciones no lineales y con funciones trascendentes la ejecución del algoritmo se encuentra con los siguientes problemas:

(a) La **quiescencia prematura** que se puede alcanzar, ya que el algoritmo se detiene antes de obtener una aproximación lo más cercana al conjunto de posibles soluciones o antes de detectar la inconsistencia.
(b) La **convergencia lenta** hacia la solución, que hace que sea prácticamente imposible llegar a una solución.
(c) La **consistencia local** de la solución obtenida hace que nos se detecten posibles inconsistencias.

Durante un tiempo estos resultados negativos fueron atribuidos a la complejidad analítica inherente a la resolución de lo sistemas de ecuaciones e inecuaciones. Los avances recientes no aparecen que lleguen a estas conclusiones y proponen remedios y mejoras considerables.

- **Algoritmos de alto grado de consistencia en dominios continuos**

En la bibliografía se han expuesto diferentes niveles de consistencia para las redes de restricciones. Así se tiene: la **consistencia de nodo**, la **consistencia de arco** y la **consistencia de camino**, definidas y generalizadas como k-consistencia y k-consistencia fuerte.

Una red es **k-consistente** si para una instanciación dada de cualquier conjunto de variables k-1, que satisfacen todas las relaciones directas entre estas variables, existe una instanciación de la variable k-ésima tal que los k valores que toma satisfacen todas las relaciones entre las k-variables.

Se puede, entonces, decir que la consistencia de arco se corresponde con la 2-consistencia y la consistencia de camino con la 3-consistencia.

Desde el punto de vista algorítmico, las técnicas de consistencia de arco han sido objeto de una particular atención y se han propuesto diferentes algoritmos como: AC-1, AC-2 y AC-3 introducidos por Mackworth [Mackworth77]. Este último algoritmo es el que se ha denominado **algoritmo de propagación**. Más tarde ha sido mejorada para dominios discretos, proponiéndose AC-4 por Mohr [Mohr86], que fue generalizado como AC-5 por Deville [Deville91] y de nuevo revisado como AC-6 por Bessiere [Bessiere93].

Para el caso de las consistencia de camino se han diseñado los algoritmos PC-1 por Mackworth [Mackworth 77], que posteriormente fue revisado como PC-2 por el mismo y mejorado para dominios

discretos como PC-3 por Mohr [Mohr86]. Los algoritmos de más alto grado de consistencia que la consistencia de arco requieren herramientas para sintetizar restricciones.

La primeras implementaciones prácticas de las técnicas de consistencia en dominios continuos usaron el **algoritmo de Waltz**, similar al algoritmo AC-3 y las operaciones básicas de la aritmética de intervalos para refinar las etiquetas intervalares. Entre dichas implementaciones cabe destacar las realizadas por Davis [Davis87].

Los algoritmos de consistencia de camino tales como PC-1 y PC-2, se aplicación con éxito en restricciones continuas de tipo temporal y espacial. Pero en el caso general, se puede decir que es difícil aún siquiera alcanzar la consistencia de arco. No obstante varios autores han investigado algoritmos de más alto grado de consistencia en dominios continuos. Entre ellos destacan el algoritmo de propagación de tolerancias de Hyvönen [Hyvönen92], los algoritmos de k-B-consistencia de Lhom [Lhomme93], los algoritmos de consistencia de caja de Benhamou [Benhamou94] y los algoritmos de consistencia basados en quad-trees (árboles 2^k) de SamHar [SamHar95]. A continuación vamos a efectuar una breve introducción a dichos algoritmos.

A. **Propagación de Tolerancias**

Con objeto de implementar el operador ***Aproximación*** se definen en este método propuesto por Hyvönen las llamadas **restricciones primitivas** como aquellas restricciones

$$c(X_1,\cdots,X_k) \qquad (10.6)$$

tales que $\forall i \in 1\cdots k$ es posible encontrar dos funciones F_i^{min} y F_i^{max} tal que, para cualquier hipercubo determinado por el dominio de las variables $I = I_1 \times\cdots\times I_k$, entonces:

$$F_i^{min}(I_1,\cdots,I_{i-1},I_{i+1},\cdots,I_k) \text{ y } F_i^{max}(I_1,\cdots,I_{i-1},I_{i+1},\cdots,I_k) \qquad (10.7)$$

son los valores mínimo y máximo de las proyecciones resultantes de aplicar dicha restricción sobre los ejes del hipercubo. A partir de ellas el operador ***Refinar*** aplicado sobre una restricción es de la forma

Algoritmo del operador *Refinar*

I':=I
Para cada $v_i \in c$
$\quad I_i' := I_i' \cap [F_i^{min}(I_1,\cdots,I_{i-1},I_{i+1},\cdots,I_k), F_i^{max}(I_1,\cdots,I_{i-1},I_{i+1},\cdots,I_k)]$
Fin Para

Este método usa inicialmente una generalización del **algoritmo de Waltz** para la consistencia local. Si se tiene una restricción $c(x_1,\cdots,x_k)$, con objeto de buscar las funciones F_i anteriores, se definen unas **funciones solución** de la forma

$$x_i = f_i(x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n) \qquad (10.8)$$

siendo f_i la función resultante de realizar las operaciones simbólicas para despejar la variable x_i de la restricción $c(x_1,\cdots,x_k)$. Si se tienen las funciones solución de una restricción c, se puede comprobar la consistencia local para una determinada variable x_i, cuyo dominio es I_i, cuando evaluamos la restricción solución para dicha variable tomando otras variables los valores de sus respectivos dominios.

Con estas ideas Hyvönen desarrolló el algoritmo de **propagación de tolerancia local**, que en síntesis realiza la construcción de una agenda de funciones solución, a partir de las restricciones del problema, y se va comprobando la consistencia local de la forma anteriormente expuesta.

Sin embargo, la aplicación de estas ideas conlleva un conjunto de inconvenientes:

(a) La definibilidad de las restricciones, puesto que ciertos valores de las variables deben excluirse de la definición de las restricciones.
(b) La aplicabilidad de las funciones solución, porque estás no pueden no estar definidas para ciertos valores de los dominios. Por otra parte, para funciones solución complicadas se hace necesario un estudio de la monotonía de las mismas para obtener valores de la solución de dichas restricciones.

La solución a estos problemas viene del estudio del espacio de definición para cada una de las funciones solución, dividiendo el espacio de definición en tantos elementos como sean necesarios para que siempre esté definida, y también estudiando los criterios de monotonía mediante las derivadas parciales. Esto produce el primer inconveniente de este método, que es la explosión combinatoria de los dominios de definición que se tienen que tratar con el algoritmo.

Pero la aportación más importante de este algoritmo es que permite generalizarlo para determinar la consistencia global a diferencia de otras aproximaciones donde esto no es posible. La consistencia global como se ha visto siempre implica la consistencia local, y cualquier solución global está recogida dentro de los resultados obtenidos mediante la aplicación del algoritmo de propagación de tolerancia. Pero éste solo consigue consistencia local y por lo tanto tiene los siguientes inconvenientes:
(a) No obtener la solución más estrecha, y por tanto pueden aparecer soluciones no consistentes globalmente.
(b) No detectar inconsistencias globales

Básicamente, el que no se consiga la consistencia global se debe a la existencia de ciclos. No obstante, hay casos de restricciones donde al no haber ciclos, la consistencia local asegura la consistencia global. Dichos casos son:

(a) Un conjunto de restricciones acíclicas
(b) Si en el conjunto de restricciones cíclicas, las variables que son cíclicas tienen valores constantes.

No obstante, tales casos se dan raras veces, por lo que Hyvönen propone dos métodos para mejorar su algoritmo de propagación de tolerancia local para conseguir la consistencia global: la **partición dinámica** y la **propagación de tolerancia global**.

La técnica de partición dinámica se basa en partir el dominio de unas de las variables del conjunto de corte, dejando los demás dominios iguales, y si una de las partes hace inconsistente la red se elimina del dominio. Después se continua con las demás variables del conjunto de corte. Esta técnica tiene dos problemas importantes, por una parte encontrar la condición de terminación necesaria para que la solución sea globalmente consistente, y por otra la eficiencia cuanto a cual es el número de veces que resulta más adecuado partir el dominio de las variables del ciclo para que el algoritmo no sea computacionalmente muy costoso.

Ante los inconvenientes anteriores Hyvönen propone la técnica de propagación de tolerancia global, cuya idea principal se basa en encontrar las llamadas **funciones solución globales**, que podemos decir que son aquellas en que si se tiene un conjunto de restricciones C con variables x_1, ... , x_k, la función solución global sería

$$x_i = CG(x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_k) \qquad (10.9)$$

que evalúa el valor actual de x_i cuando los valores de las restantes variables varían independientemente dentro de sus tolerancias. Esto se realiza exclusivamente para aquellas variables que pertenecen al conjunto de corte. Podemos decir que lo que se hace es abstraer el conjunto de restricciones en una única restricción global para dicha variable.

B. **Técnicas de Arco-K-B-Consistencia**

La arco-B-consistencia propuesta por Lhomme [Lhomme93] consiste en considerar solamente los límites de los dominios. Esta técnica consigue refinar la solución mediante partición y refutación. Si suponemos una variable de un problema X con un dominio $I_x = [a,b]$, probaremos a restringir de acuerdo con los

algoritmos de propagación dicho dominio incrementado su extremo inferior. Sea el punto $c \in (a,b)$. Entonces, se añade la restricción $X \in [a,c)$. Si la aplicación de los anteriores algoritmos detecta una contradicción no podemos asegurar nada, pero podemos repetir el proceso con un punto con un punto c' que esté más cercano a y repetir el proceso. Igual puede hacerse con el extremo superior del dominio de la variable. Esta es una técnica que permite conseguir soluciones más estrechas con respecto a las ya encontradas por las anteriores técnicas. Los métodos de partición pueden ser de varios tipos, pero una posibilidad es partir el dominio en dos, y de nuevo cada parte en dos, etc.

C. Técnicas de Consistencia de Caja

La consistencia de caja es una aproximación de la consistencia de arco, ya que este tipo de consistencia no puede ser automatiza en general cuando se trabaja con números reales y restricciones complejas. Por ello se han propuesto diferentes tipos de consistencia, cuya idea fundamental es obtener la consistencia local de manera más eficiente.

Se dice entonces que: una proyección de una restricción <c,i> es **consistente de arco** con respecto a $I_1 \times \cdots I_n$ si se cumple

$$I_i = I_i \cap \Pi_{x_i} c \equiv \Pi_{x_i} (c \cap I_i) \qquad (10.10)$$

La consistencia de arco no puede calcularse eficientemente en restricciones complejas, debido a las limitaciones de la representación de los números reales en los ordenadores. Por tanto, es necesario aproximar este concepto con otros que alcance un equilibrio entre su poder de reducción, la complejidad de las restricciones primitivas y la eficiencia computacional. Se han expuesto en la bibliografía diferentes tipos de consistencia que desde la más fuerte a la más débil son las siguientes:

(a) **Consistencia intervalar**. Una proyección de una restricción <c,i> es **consistente intervalar** con respecto a $I_1 \times \cdots I_n$ si se cumple

$$I_i = \boldsymbol{Aprox}_U (I_i \cap \Pi_{x_i} c) \qquad (10.11)$$

La idea que subyace en esta nueva consistencia es adaptar la consistencia de arco a la precisión de la máquina, aproximando los números reales por números en coma flotante. El problema que presenta es se deben manejar una gran cantidad de intervalos y su tratamiento puede llegar a ser computacionalmente muy complejo. Entonces se propone una nueva definición que llene los espacios entre intervalos separados.

(b) **Consistencia por envolvente de una restricción proyectada**. Una proyección de una restricción <c,i> es **consistente por envolvente** con respecto a $I_1 \times \cdots I_n$ si se cumple

$$I_i = \boldsymbol{Aprox}_E (I_i \cap \Pi_{x_i} c) \qquad (10.12)$$

Tanto la consistencia intervalar como la consistencia de envolvente son adecuadas cuando se trabaja con restricciones primitivas, que no presentan múltiples ocurrencias de una variable, pero son demasiado exigentes cuando las restricciones son complejas, puesto que pueden requerir explorar múltiples combinaciones de los intervalos que aparecen en la restricción. Por ello, se hace necesario definir una consistencia menos exigente.

(c) **Consistencia de caja**. Una proyección de una restricción <c,i> es **consistente por envolvente** con respecto a $I_1 \times \cdots I_n$ si se cumple

$$I_i = EC(I_i \cap \{a_i \in R | \overline{c}(I_1, \cdots, I_{i-1}, \boldsymbol{Aprox}(a_i), I_{i+1}, \cdots, I_n) \qquad (10.13)$$

donde $\overline{c}$ es la extensión intervalar de c. El concepto de extensión intervalar de una restricción se expondrá más adelante. Hay casos donde las restricciones son a la vez consistente por envolvente y consistente de caja, como por ejemplo en las primitivas. Las diferencias más imperantes se encuentran cuando tenemos restricciones con múltiples ocurrencias para una variable. Se puede decir que la consistencia de caja viene determinada por la extensión intervalar que use para el cálculo. Para una extesión intervalar EI(c) la definición de consistencia de caja es equivalente a la siguiente definición: Una proyección de una restricción <c,i> es **consistente por envolvente** con respecto a $I_1 \times \cdots I_n$ y un operador extensional EI si se cumple

$$EI(c)(I_1,\cdots,I_{n-1},[l,l+\varepsilon],I_{i+1},\cdots,I_n) \wedge EI(c)(I_1,\cdots,I_{n-1},[u,u-\varepsilon],I_{i+1},\cdots,I_n) \quad (10.14)$$

siendo *l* el valor izquierdo del intervalo, *u* el valor derecho y ε un valor positivo suficientemente pequeño. Entonces se puede decir que: Una restricción es consistente de caja con respecto a I y EI si cada una de sus proyecciones es consistente de caja con respecto a I y EI. Un sistema de restricciones es consistente de caja con respecto a I y EI si cada restricción es consistente de caja con respecto a I y EI. Por lo tanto, se puede decir, de forma intuitiva que la condición anterior establece que los intervalos no pueden ser podados más usando dicho operador extensional. La consistencia de caja puede ser reducida a dos subproblemas que consisten en estrechar el intervalo correspondiente usando una variación del método de Newton intervalar. Estas técnicas consiguen la consistencia local con intervalos más anchos que la consistencia de arco, pero con una eficiencia computacional mejor.

D. **Técnicas de Consistencia basada en Quad-Trees** (árboles 2^k)

Como consecuencia de los problemas ya discutidos de los algoritmos de propagación de restricciones con intervalos se ha propuesto un nuevo algoritmo para restricciones binarias. Dicho algoritmo está basado en la consistencia de arco y permite mediante una regla de propagación apoyada en la identificación y clasificación de extremos locales, obviar algunos de los problemas que el algoritmo de propagación presentaba. Dichos trabajos desencadenaron en trabajos posteriores de SamHar [SamHar95] que pretenden alcanzar la consistencia global mediante la utilización de quad-trees, un tipo de representación usada comúnmente en visión por ordenador y tratamiento de imagen, para la representación de las regiones que son soluciones posibles. En estos trabajos se muestran algoritmos para procesar la restricciones y espacios de solución de complejidad arbitraria, pero con una resolución máxima prefijada. Los métodos anteriores aproximaban las soluciones mediante hipercubos, pero el espacio de soluciones era la mayoría de veces demasiado ancho, introduciéndose demasiadas soluciones espúreas. Sin embargo, en este método la región de solución $R_{x_1,\cdots,x_n}$ puede aproximarse por una descomposición jerárquica de su espacio de solución en árboles 2^k, donde el valor k depende del tipo de consistencia que se propone para el problema. Este método aunque es muy general, tiene el inconveniente que resolver cualquier problema está delimitado por el tiempo en que la resolución se considera aceptable. Se comprueba que la complejidad polinomial es solamente válida cuando se cumplen ciertas condiciones de convexidad en las restricciones, y se garantiza que el algoritmo termina con una correcta solución que no depende de la forma de las restricciones. Esto es una mejora significativa sobre las técnicas anteriores que utilizan las restricciones mediante métodos iterativos que se encuentran abocados a considerar los problemas de convergencia y estabilidad que hacen que no se pueda garantizar la terminación del algoritmo. Pero el precio que se paga en este trabajo es la discretización que se realiza, que dependiendo del tipo de problema puede ser demasiado grande.

10.2 Problemas de Satisfacción de Restricciones Intervalares

Una vez presentada la problemática general de los problemas de satisfacción de restricciones, vamos a profundizar en tipo particular de problema para el caso en que las restricciones son intervalos.

- **Sintaxis**

Un **problema de satisfacción de restricciones intervalares** (ICSP) se puede representar de la siguiente forma:

$$\begin{aligned} &E_1,\cdots,E_n \\ &P_1:=X_1,\cdots,P_m:=X_m \end{aligned} \quad (10.15)$$

donde E_i son ecuaciones, P_j son variables (explícitas) utilizadas en las mismas y X_j son intervalos reales cerrados: $[a,b]=\{x|a\le x\le b\}$. $P:=X$ asigna el intervalo X a la variable P. Si alguna de las variables P no se le asocia valor, se supone por defecto que $P:=[-\infty,\infty]$.

Un par de ejemplos de problema de este tipo serían:

Ejemplo 1:

$$F = C*1.8+32$$
$$C:=[1,5]$$
$$F:=[27,35]$$

Ejemplo 2:

$$Z = X*Y$$
$$e^X + Z = -0.2$$
$$\sin(-Z)+X+Y=-0.4$$

Las funciones anidadas $y = f(x_1,\cdots,x_n)$ se pueden representar de forma equivalente como relaciones de tipo restricción $\{\langle x_1,\cdots,x_n,y\rangle\}$. Por lo tanto, $E_1,\cdots,E_n$ puede ser fácilmente descompuesto en un conjunto equivalente de restricciones simples, **la red de restricciones**, representado cada función $f(x_1,\cdots,x_n)$ por un única variable (interna), y utilizando *y* en la restricción huésped en lugar de $f(x_1,\cdots,x_n)$.

Para el ejemplo 2, presentado anteriormente la descomposición que se propone es la siguiente:

$$\{X*Y=Z,\quad e^X=X_1,\quad X_1+Z=-0.2,\quad -Z=X_2,\quad \sin(X_2)=X_3,\quad X_3+X=X_4,\quad X_4+Y=-0.4\}$$

donde X_i son las variables que se han introducido para realizar la descomposición de las restricciones en restricciones simples.

- **Semántica**

La **situación de tolerancia**

$$\{P_1:=X_1,\cdots,P_m:=X_m\} \qquad (10.16)$$

en un problema de satisfacción de restricciones intervalares (ICSP) se refiere al conjunto de situaciones de valor exacto

$$\{P_1:=x_1,\cdots,P_n:=x_n \mid x_j \in X_i, i=1,\cdots,n\} \qquad (10.17)$$

a las que denominamos **extensiones**.

Una relación de restricción $\{<x_1,\cdots,x_n>\}$ correspondiente a una función $x_n = f(x_1,\cdots,x_{n-1})$ es **satisfecha** por una extensión, si y sólo si, se cumple $x_n = f(x_1,\cdots,x_{n-1})$ dentro de la misma.

Un problema de satisfacción de restricciones intervalares (ICSP) es **admisible**, si y sólo si, su situación de tolerancia posee una extensión que satisfaga todas las restricciones, es decir, las ecuaciones tienen una solución dentro de los intervalos:

"$\exists\{P_1:=x_1\in X_1,\cdots,P_m:=x_m\in X_m\}$
tal que se satisfacen todas las restricciones E_1, ... , E_n"

Por ejemplo, el Ejemplo 1 presentado anteriormente, presenta un problema ICSP admisible porque $C:=1\in[1,5]$ y $F:=33.8\in[27,35]$ satisfacen la única restricción que posee.

Una variable $P_i:=X_i$ es **consistente**, si y sólo si, todas las instancias $P_i:=x, x\in X_i$, pueden ser satisfechas respecto a todas las restricciones por alguna extensión:

"$\forall x\in X_i \quad \exists\{P_1:=x_1\in X_1,\cdots,P_i:=x,\cdots,P_m:=x_m\in X_m\}$

tal que se satisfacen todas la restricciones $E_1,..., E_n$"

De forma intuitiva, una tolerancia no debe contener valores "extra" que no puedan ser satisfechos por los intervalos dados.

Una situación de tolerancia es **globalmente consistente**, es decir, es una **solución de tolerancia** (global), si y sólo si, cada una de sus variables es consistente. Una variable es **localmente consistente**, si y sólo si, es consistente respecto a todas las restricciones a las que está conectada directamente. La consistencia local de una situación (solución) de tolerancia, significa que todas las variables son localmente consistentes. Por ejemplo, el Ejemplo 1 es global y localmente inconsistente, porque $C:=4 \in [1,5]$ no puede ser satisfecho por ningún $F \in [27,35]$.

Una situación de tolerancia

$$S = \{P_1 := X_1, \cdots, P_n := X_n\} \qquad (10.18)$$

es más **general** que

$$S' = \{P_1 := X'_1, \cdots, P_n := X'_n\} \qquad (10.19)$$

si y sólo si, $X_i' \subseteq X_i$ para $i = 1, \cdots, n$, y se indica mediante $S' \subseteq_s S$.

La relación de generalidad $\subseteq_s$ es reflexiva, antisimétrica y transitiva, de donde, podemos afirmar que una ordenación parcial en el conjunto de las situaciones de tolerancia de un ICSP. De forma intuitiva, las situaciones forman una jerárquica de generalidad definida por $\subseteq_s$. Por ejemplo, parte de la relación $\subseteq_s$ para el Ejemplo 1 se presenta en la Fig. 10.5, con las relaciones más generales en la parte superior de la retícula. Las situaciones consistentes se presentan en negrita frente a la situaciones inconsistentes que se presentan en letra normal. Las situaciones inadmisibles se presentan en cursiva. Las situaciones inconsistentes puede ser admisible o inadmisibles.

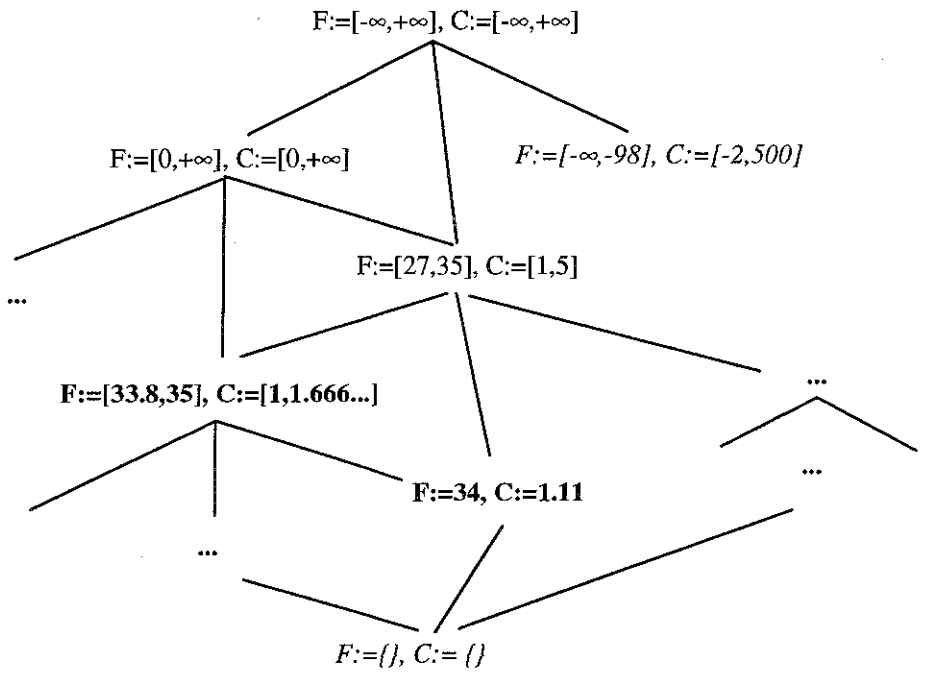


Fig. 10.5. Retícula de situaciones para la restricción F = C * 1.8 + 32. La consistencia se indica con negritas mientras que la inconsistencia se indica en letra normal. La inadmisibilidad se indica en cursiva.

Matemáticamente, la estructura resultante es una retícula ("lattice") $L=\{S,\cup_S,\cap_S\}$, donde $\cup_S$ representa la unión, mientras que $\cap_S$ representa la intersección de situaciones S definidas, respectivamente, de la siguiente manera:

$$\{P_i:=X_i\}\cup_S\{P_i:=Y_i\}=\{P_i:=X_i\cup Y_i\} \quad \text{para } i=1,\cdots,n \qquad (10.20)$$

$$\{P_i:=X_i\}\cup_S\{P_i:=Y_i\}=\{P_i:=X_i\cup Y_i\} \quad \text{para } i=1,\cdots,n \qquad (10.21)$$

La menor cota superior de las soluciones de un problema de satisfacción de restricciones intervalares (ICSP) se denomina la **solución común general menor (LGCS)**. La LGCS generaliza las soluciones numéricas exactas S_j de un ICSP, así como las soluciones de tolerancia más generales mediante:

$$LGCS=\cup_S\{S_j\} \qquad (10.22)$$

Los valores de las variables en la LGCS no se pueden representar siempre como intervalos continuos. Si $P:=X_1$ en la solución S_1, $P:=X_2$ en la solución S_2, y $X_1\cap X_2=\{\}$, entonces $P:=X_1\cup X_2$ es un intervalo discontinuo en $LGCS=S_1\cup_S S_2$. Por ejemplo, la LGCS del problema

$$X^2=Y$$
$$Y:=[4,9]$$

es

$$\{X:=[-3,2]\cup[2,3],\ Y:=[4,9]\}.$$

A la colección de conjuntos $D=X_1\cup\cdots\cup X_n$ de intervalos mútuamente excluyentes se denomina **división** y se representa como

$$D=\left[x_{1,1},x_{1,2}\middle|x_{2,1},x_{2,2}\middle|\cdots\middle|x_{n,1},x_{n,2}\right] \qquad (10.23)$$

donde $X_i=\left[x_{i,1},x_{i,2}\right]$ para i = 1, ... , n y $x_{i,2}<x_{i+1,1}$ para i = 1, ... , n-1. A los intervalos X_i se los denomina **constituyentes** de la división D. Por simplicidad, la constituyencia se indicará mediante $X_i\in D$, aunque la división D es en realidad un conjunto de valores reales. Un intervalo formado por un único valor (singleton) [x,x] constituyente de una división D se representa como el valor exacto x. Por ejemplo:

$$[-\infty,0^-]\cup[0,0]\cup[0^+,1]=\left[-\infty,0^-\middle|0,0\middle|0^+,1\right]=\left[-\infty,0^-\middle|0\middle|0^+,1\right]$$

donde x^- y x^+ representan, respectivamente, un número ligeramente menor y mayor que x.

Mediante la generalización de la noción de intervalo mediante la noción de división, cualquier conjunto de soluciones de un ICSP se puede representar mediante un único LGCS. Aunque siempre podemos continuar utilizando intervalos continuos como caso especial, si es necesario. La idea de utilizar divisiones en lugar de intervalos contrasta con otros enfoques utilizados en trabajos como el de Davis [Davis87]. El problema que aparece cuando sólo se utilizan intervalos se debe a que se deben generar conjuntos de soluciones intervalares, lo que conduce a un gran número de soluciones que pueden hacer que el problema sea computacionalmente intratable.

- **Solución**

La formulación que se había dado en la sección anterior de un problema de satisfacción de restricciones intervalares se puede reescribir de la siguiente manera en términos de la retícula de soluciones:

"Dado un problema de satisfacción de restricciones intervalares (ICSP), su solución consiste en determinar su solución común general menor (LGCS)"

Por ejemplo, el Ejemplo 1 presenta el problema determinar los intervalos (divisiones) mútuamente consistentes con la restricción

$$F = C * 1.8 + 32$$

$$C := [1,5]$$

$$F := [27,35]$$

Un algoritmo de solución de dicho tipo de problemas debería detectar las inconsistencias y refinar las tolerancias siguiendo la retícula de la Fig. 10.5 hasta llegar a la solución $\{C := [1,1.666...], F := [33.8,35]\}$. Si se continuara refinando las tolerancias eliminaríamos extensiones numéricas de la ecuación: la LGCS es la solución más ajustada que generalizada todas las soluciones de un problema ICSP.

10.3 El Algoritmo de Waltz

Fue el primer algoritmo propuesto para resolver redes de restricciones intervalares. Dicho algoritmo refina cada restricción, utilizando el operador ***Refinar*** sobre cada una de las restricciones y nodos de la red hasta que ya no consigue mejorar la solución.

El operador ***Refinar*** utilizado por el algoritmo de Waltz se define de la siguiente manera: sea C una restricción sobre los nodos $X_1,...,X_k$. Sea S_i el intervalo asociado al nodo X_i. Entonces

$$\textbf{\textit{Refinar}}(C,X_j) = \{a_j \in S_j | \exists (a_i j \in, S_i, i = 1,\cdots,k, i \neq j) \quad C(a_1,\cdots,a_j,\cdots,a_k)\} \qquad (10.24)$$

o sea, ***Refinar***(C,X_i) es el conjunto de valores para X_j que son consistentes con la restricción C y con todos los valores del intervalo S_i. Un valor a_j se encuentra en el conjunto ***Refinar***(C,X_i) si a_j forma parte de Sj y de la k-tupla $a_1,...,a_k$ que satisface C y todos los valores del intervalo S_i.

El operador ***Refinar***(C,X_i) es consistente en el sentido de que si una tupla satisface la restricción y los intervalos iniciales para cada una de las variables que la forman, entonces la tupla refinada obtenida al aplicar el operador Refinar también satisface el intervalo refinado. Así mismo, puesto que dicho operador es consistente y que el algoritmo de Waltz es simplemente una iteración de llamadas al operador ***Refinar*** es también consistente, de forma que dado un conjunto de valores que satisfacen todas las restricciones y los intervalos iniciales, entonces dicho conjunto de valores también satisfacerá los intervalos determinados por el algoritmo de Waltz.

El algoritmo de Waltz tiene la siguiente estructura:

Inicialización de una lista L con todas las restricciones.
Mientras la lista L no este vacía
    Extraer restricción C de la lista L.
    Aplicar el operador ***Revisar*** sobre C.
    Para cada intervalo X_i refinado
        Para cada restricción $C' \neq C$ que utilice X_i
            Añadir C' a la lista L
        Fin Para
    Fin Para
Fin Mientras

En dicho algoritmo se utiliza el operador ***Revisar*** para refinar todos los intervalos asociados a una determinada restricción, devolviendo el conjunto de intervalos modificados.

El algoritmo para el operador ***Revisar*** tiene la siguiente estructura:

Inicialización de la lista C de intervalos modificados por el operador.
Para cada intervalo X_i que forma parte de la restricción C
Aplicar el operador ***Refinar*** sobre dicha restricción C e intervalo X_i
Si el intervalo refinado es el intervalo nulo
Entonces el algoritmo finaliza debido
a la inconsistencia de las restricciones
En caso contrario si $S \neq S_i$ entonces
$S_i = S$
Añadir el intervalo X_i a la lista de intervalos de C
modificados por el operador ***Revisar***
Fin Si
Fin Para

Veamos un ejemplo de aplicación del algoritmo de Waltz sobre un red de restricciones concreta:

$$x + y = z$$

$$y \leq z$$

$$x \in [1,10] \quad y \in [3,8] \quad z \in [2,7]$$

Dicha red de restricciones se puede representar de la siguiente manera:

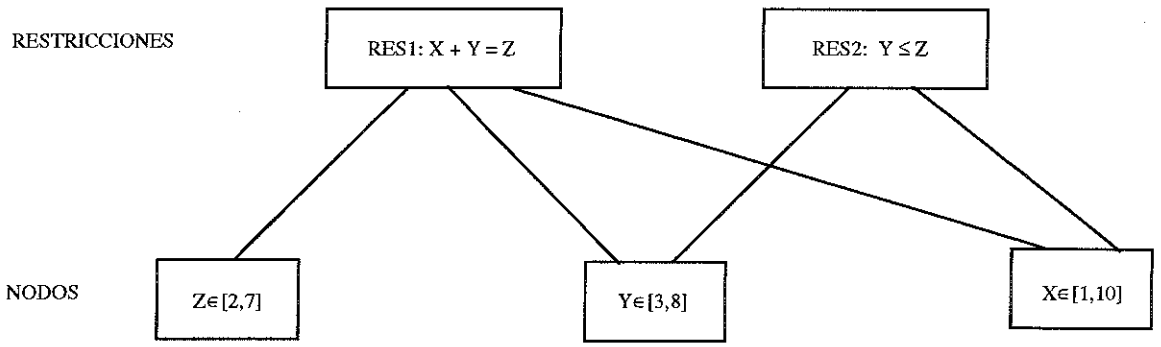


Fig. 10.6 Red de restricciones asociada al ejemplo resuelto mediante el algoritmo de Waltz

El algoritmo de Waltz aplicado a este ejemplo procederá de la siguiente manera: la lista de restricciones inicial L contendrá las dos restricciones de la red: RES1 y RES2. A continuación la restricción RES1 () será extraída, y puesto que:

(a) $x \geq 1$ e $y \geq 3$, entonces $z \geq 4$, de forma que las cotas sobre z se modificarán hasta [4,7].
(b) $z \leq 7$ e $y \geq 3$, entonces $x \leq 4$, de forma que las cotas sobre x se modificarán hasta [1,4].
Puesto que x, z han cambiado sus cotas, añadiremos la restricción RES2 a la lista.

A continuación la restricción RES2 ($y \leq x$) será extraída de la lista, y puesto que:

(a) $x \leq 4$, entonces $y \leq 4$, de forma que las cotas sobre y se modificarán hasta [3,4].
(b) $y \geq 3$, entonces $x \geq 3$, de forma que las cotas sobre x se modificarán hasta [3,4].
Puesto que x, y han cambiado sus cotas, añadiremos la restricción RES1 a la lista.

A continuación la restricción RES1 (x+y=z) será extraída de la lista, y puesto que:

(a) $x \geq 3$ e $y \geq 3$, entonces $z \geq 6$ de forma que las cotas sobre z se modificarán hasta [6,7].
Puesto que sólo z ha cambiado sus cotas y no existe ninguna otra restricción a parte de RES1 que tenga a z como variable, entonces no se añadirá ninguna restricción a la lista.

Finalmente, debido a que ya no existen más restricciones a procesar en la lista el algoritmo finalizará.

10.4 Propagación de la Tolerancia

En primer lugar, repasaremos la noción de función intervalar, para a continuación presentar un algoritmo de propagación intervalar local.

- **Funciones intervalares básicas**

Tal como hemos visto en capítulos anteriores la aritmética o álgebra intervalar (IA) trabaja con **intervalos** cerrados $X=[a,b]=\{x|a\leq x\leq b\}$. El valor $\frac{a+b}{2}$ se denomina el **centro** mientras que el valor b-a se denomina la **anchura** del intervalo X. El álgebra de intervalos es una generalización del álgebra de valores exactos ordinaria: cualquier número real x se puede representar como un intervalo de un único valor [x,x].

El álgebra de intervalos, también, generaliza las funciones reales $f(x_1,\cdots,x_n)$ mediante las correspondientes funciones intervalares $F(X_1,\cdots,X_n)$ de acuerdo con

$$F(X_1,\cdots,X_n)=\{f(x_1,\cdots,x_n)\,|\,x_i\in X_i \quad \text{para } i=1,\cdots,n\} \qquad (10.25)$$

Las variables en mayúscula se utilizan para indicar que son variables intervalares. De forma intuitiva, se puede ver que $F(X_1,\cdots,X_n)$ evalúa el rango de valores de la función f cuando $x_1,\cdots,x_n$ toman valores **independientemente** dentro de los correspondientes intervalos. Así, por ejemplo las funciones intervalares racionales se pueden definir la siguiente manera:

$$\begin{aligned} A+B&=\{a+b\,|\,a\in A, b\in B\}\\ A-B&=\{a-b\,|\,a\in A, b\in B\}\\ A*B&=\{a*b\,|\,a\in A, b\in B\}\\ A/B&=\{a/b\,|\,a\in A, b\in B\} \quad \text{si } 0\notin B \end{aligned} \qquad (10.26)$$

La generalización intervalar de otras funciones básicas como X^n, exp(X), sin(X) puede también definirse fácilmente.

Desde el punto de vista computacional, utilizaremos las definiciones de dichas funciones intervalares que nos proporcionen directamente los límites de los intervalos resultado. Así, por ejemplo, las definiciones anteriores pueden reescribirse de la siguiente manera:

$$\begin{aligned} [a,b]+[c,d]&=[a+c,b+d]\\ [a,b]-[c,d]&=[a-d,b-c]\\ [a,b]*[c,d]&=[\min(ac,ad,bc,bd),\max(ac,ad,bc,bd)]\\ [a,b]/[c,d]&=[a,b]*[1/d,1/c] \quad \text{si} \quad 0\notin[c,d] \end{aligned} \qquad (10.27)$$

En el álgebra de intervalos, una función intervalar F se denomina una **extensión** de su restricción real f si $F(X_i,\cdots,X_n)=f(x_1,\cdots,x_n)$ para todos los argumentos real. Para funciones anidadas complicadas f,

es a menudo difícil o incluso imposible definir una extensión intervalar exacta, aunque siempre se pueden encontrar extensiones que cumplan

$$F(X_1,\cdots,X_n) \supseteq \{f(x_1,\cdots,x_n \mid x_i \in X_i \text{ para } i=1,\cdots,n\} \qquad (10.28)$$

que proporcionan intervalos mayores que el rango de valores real de la función f.

- **Propagación intervalar local**

Cualquier función exacta $x_n = f(x_1,\cdots,x_{n-1})$ se puede definir de forma equivalente mediante la relación restricción $REL(f) = \{< x_1,\cdots,x_n > \mid x_n = f(x_1,\cdots,x_{n-1})\}$. En la propagación intervalar local, las condiciones de consistencia local de la restricción f se definen mediante **funciones solución simétricas**

$$F_i(X_1,\cdots,X_{i-1},X_{i+1},\cdots,X_n) = \{x_i \mid < x_1,\cdots,x_i,\cdots,x_n > \in REL(f), x_j \in X_j \text{ para } i=1,\cdots,n\}$$

donde $F_i(X_1,\cdots,X_n)$ evalúa el intervalo de todos los posibles valores cuando el resto de variables varían independientemente dentro de sus tolerancias. Debido a que esto es lo que realizan las funciones intervalares del álgebra de intervalos, las funciones solución se pueden obtener algebraicamente a partir de las expresiones de las restricciones. Así, por ejemplo, las funciones solución de la restricción $X+Y=Z$ son $X=Z-Y$, $Y=Z-X$ y $Z=X+Y$.

Si las funciones solución de una restricción F son $Y_1 = F_1(X_1,\cdots,X_n),\cdots,Y_n = F(X_1,\cdots,X_n)$, entonces, utilizando la definición de consistencia, las variables son localmente consistentes con respecto a F, sí y sólo si:

$$Y_1 \subseteq F_1(X_1,\cdots,X_n),\cdots,Y_n \subseteq F_n(X_1,\cdots,X_n) \qquad (10.29)$$

La condición para asegurar la consistencia local en un problema ICSP se obtiene cuando se cumple la condición enunciada anteriormente para cada una de las restricciones del problema: una situación es localmente consistente, si y sólo si, todas las funciones solución del problema ICSP evalúan intervalos mayores ($\subseteq$) que el valor de sus variables.

A continuación se presenta un algoritmo que se puede utilizar para determinar la LGCS local. El algoritmo utiliza como entrada las funciones solución del problema ICSP y los valores iniciales de las tolerancias de las variables. El algoritmo elimina los valores imposibles mediante la operación intersección presentada en el apartado anterior hasta que se cumpla $Y \subseteq F(X_1,\cdots,X_n)$ para todas las funciones solución.

- **Algoritmo de Propagación de Tolerancia Local**

El algoritmo de propagación de tolerancia local tiene los siguiente pasos:

Algoritmo de Propagación de Tolerancia Local

Paso 1. Inicialización de la *Agenda* con las funciones solución de las restricciones del problema ICSP (en algún orden).

Paso 2. Inicialización de *S* con el valor inicial de la tolerancia para las variables del problema ICSP.

Paso 3. Para cada función solución *X := F(...)* de la Agenda hasta que esta este vacía, *Agenda = {}* realizaremos las siguientes operaciones:

(a) Evaluaremos la siguiente función solución de la Agenda, *X':= F(...)*

(b) Si *X'* es igual al valor de *X* en *S*, entonces eliminaremos de la *Agenda* todas las funciones solución que provengan de la misma restricción que F(...), y volveremos al paso 3. En caso contrario *Int* := *X'* ∩ *(el valor de X en S)*
(c) Si Int = { }, entonces devolveremos como solución { }. La red es inconsistente, por lo tanto no existe solución.
(d) Si $X \subseteq X'$ entonces eliminaremos X := F(...) de la Agenda. En caso contrario, haremos *X:=Int* en *S*, *Agenda := Agenda* ∪ *{Funciones Solución que utilicen X} - {X:=F(...)}*.

Paso 4. Volveremos al paso 3.

Por ejemplo, consideremos el Ejemplo 1 utilizado anteriormente:

$$C*1.8 = X, \quad X+32 = F,$$

$$C:=[1,5], \quad X:=[-\infty,\infty], \quad F:=[27,35]$$

Las funciones solución correspondientes a las restricciones son:

Restricción	Funciones Solución
C*1.8=X	X=C*1.8, C=X/1.8
X+32=F	F=X+32, X=F-32

La agenda inicial de funciones solución de esta red de restricciones es, por ejemplo (cualquier ordenación de dichas funciones es posible):

$$X = C*1.8, \quad X = F-32, \quad C = X/1.8, \quad F = X+32$$

En la tabla que se presenta a continuación muestra como el algoritmo de propagación intervalar funciona sobre el ejemplo que acabamos de presentar. Los valores finales que se obtienen para las variables son:

$$C:=[1,1.666...], \quad X:=[1.8,3], \quad F:=[33.8,35]$$

- **Comparación con el Algoritmo de Waltz**

Ambos enfoques se pueden utilizar para obtener soluciones locales consistentes similares. La principal diferencia entre ambos enfoques es que el algoritmo de propagación de la tolerancia las funciones solución de las restricciones se utilizan en la agenda mientras en el algoritmo de Waltz se utilizan directamente las funciones restricción. El algoritmo de propagación de la tolerancia tiene la siguientes beneficios: en primer lugar, las condiciones de consistencia se pueden definir en términos de funciones solución intervalares. En segundo lugar, el control de la propagación es más flexible: el orden de la evaluación de las funciones se puede redefinir después de cada paso. En tercer lugar, el procedimiento utilizado por el algoritmo de propagación de la tolerancia es un sistema basado en reglas del tipo "forward chaining", en el cual las reglas son las funciones solución. Esto sugiere que las técnicas de compilación de las reglas y las técnicas para generar deducciones utilizadas en los sistemas de valores exactos se puede aplicar en la propagación de la tolerancia. Y finalmente, el algoritmo local de propagación de la tolerancia local se puede generalizar a fin de obtener soluciones globalmente consistentes de una forma que no es posible mediante el algoritmo de Waltz.
Un inconveniente que presentan ambos algoritmos cuando trabajan sobre dominios continuos es que fácilmente pueden caer en un bucle infinito, incluso para problemas ICSP muy simples. Por ejemplo, consideremos el siguiente problema ICSP:

$$Y = 2 * X$$

$$X = Y$$

$$X := [0, \infty]$$

$$Y := [0, \infty]$$

Los valores de X e Y son biseccionados de forma continua y la situación sólo converge de forma asintótica hacia la solución $\{X := 0, Y := 0\}$. Ejemplos de problemas ICSP con convergencia asintótica constituyen un desafío para desarrollar técnicas que permitan representar los problemas ICSP de una forma computacionalmente más eficiente y para desarrollar estrategias para la gestión de la agenda en el algoritmo de propagación de la tolerancia o en el filtro de Waltz. En la práctica, este problema se puede resolver definiendo un nivel de precisión por debajo del cual dos intervalos se consideran iguales, dando por finalizado el algoritmo. En cualquier caso, la precisión del ordenador con el que estemos trabajando establece dicho nivel. Los problemas de errores de redondeo se pueden manejar con el álgebra de intervalos de una forma muy adecuada: redondeando los límites inferior por abajo y los límite superiores por arriba nunca se pierden las soluciones. Mientras que en sistemas que trabajan con valores exactos nunca se puede garantizar dicha seguridad.

- **Tratabilidad de la propagación intervalar**

Las técnicas de consistencia local se pueden aplicar sobre problemas de satisfacción de restricciones discretas de forma eficiente. Por ejemplo, la consistencia de tipo arc se puede obtener con un tiempo lineal con respecto al número de restricciones binarias. En el caso intervalar, sin embargo, es fácil formular problemas de satisfacción de restricciones intervalares que consuman mucho tiempo de computación y que converjan sólo de forma asintótica hacia la solución. Por lo tanto, es difícil estimar la complejidad en tiempo de cálculo de este tipo de problemas. Dicha complejidad depende fuertemente de la precisión deseada. En términos de complejidad en tiempo teórica, los sistemas que razonan con restricciones intervalares se comportan mal. Afortunadamente, evaluaciones empíricas han demostrado que en la práctica funcionan de forma rápida y efectiva sobre problemas de tamaño razonable.

- **Ejemplo de aplicación**

Vamos a aplicar la propagación local a un problema de diseño. La tarea consiste en distribuir el tiempo de trabajo que deben invertir n investigadores que deben dedicar a m proyectos. Por ejemplo, supongamos que tres investigadores a, b y c con capacidades de dedicación en horas A, B y C deben ser distribuidos en tres proyectos p_1, p_2 y p_3 que requieren una dedicación de recursos P_1, P_2 y P_3 horas respectivamente. La variable XY indica el tiempo de trabajo que un investigador con la capacidad X dedica a un proyecto que requiere Y recursos. Por ejemplo, AP_1 es el tiempo asignado al investigador a en el proyecto p_1. Las restricciones algebraicas para dicho problema son las siguientes:

$$A + B + C = P_1 + P_2 + P_3 = TOTAL \quad \text{Total de horas trabajadas}$$

$$\left.\begin{aligned} AP_1 + BP_1 + CP_1 &= P_1 \\ AP_2 + BP_2 + CP_2 &= P_2 \\ AP_3 + BP_3 + CP_3 &= P_3 \end{aligned}\right\} \quad \text{Dedicación a los Pr oyectos}$$

$$\left.\begin{aligned} AP_1 + AP_2 + AP_3 &= A \\ BP_1 + BP_2 + BP_3 &= B \\ CP_1 + CP_2 + CP_3 &= C \end{aligned}\right\} \quad \text{Dedicación de los Investigadores}$$

El problema se puede representar como una hoja de cálculo en la cual AP_i, BP_i y CP_i para i = 1,2,3 se utilizan como variables de entrada y A,B,C, P_1, P_2 y P_3 son las variables de salida que se desean calcular. Sin embargo, mediante las hojas de cálculo actuales o mediante sistemas de propagación de valores exactos los cálculo aritméticos no se pueden realizar a menos que el problema esté perfectamente

restringido. En este caso no es posible: al principio de la solución del problema, la distribución de la dedicación de la mayoría de los investigadores es más o menos desconocida y debe ser fijada de forma simultánea. Los valores de las variables inciertas no se pueden representar en absoluto mediante valores exactos. Es muy difícil ver cuáles son las combinaciones de las variables de entrada para las cuáles se podrían calcular un conjunto de parámetros de salida deseados. Un cambio en cualquier variable puede afectar cualquiera de los otros valores de las variables. Es bastante difícil para el usuario seguir la pista de todas las posibles interacciones.

Utilizando técnicas de valores exactos se llega a tener que utilizar una tediosa iteración donde se deben probar los valores de las variables de entrada, calcular los valores de las variables de salida y los resultados evaluados respecto a las restricciones del problema hasta que se llega a una asignación de variables satisfactoria. Los esquemas de relajación, que se utilizan para automatizar este tipo de iteración, son difíciles de aplicar en situaciones como esta donde el número de valores a probar es elevado. Además, sólo se puede encontrar una solución, dicha solución es exacta y no puede reflejar la inexactitud de los datos de entrada.

Los algoritmos de resolución de este tipo problemas basados en la propagación de la tolerancia pueden solucionar estos problemas. Inicialmente, el usuario establece los intervalos para las variables tan anchamente o tan estrechamente como desee. Si el usuario sugiere asignaciones de intervalos inconsistentes, el sistema puede refinar los intervalos hasta determinar la solución común general menor para el problema. La resolución del problema continua no por iteración sino de forma "top-down": el problema se va refinando paso a paso por el usuario y por el sistema hasta que se llega a determinar una solución satisfactoria. Por ejemplo, los intervalos iniciales podrían ser:

$$\{AP_i := BP_i := CP_i := A := B := C := [0,160],$$
$$P_i := TOTAL := [0,480]\},$$
$$i = 1,2,3.$$

Si el usuario establece que $P_2 := [160,480]$, entonces la propagación local de la tolerancia puede deducir los siguientes refinamientos $P_1 := [0,320]$, $P_3 := [0,320]$ y $TOTAL := [160,480]$. Aceptado dicho resultado y restringiendo más el problema diciendo que A no debería trabajar en el proyecto P_1 menos de 120 horas, o sea, $AP_1 := [120,160]$, entonces la propagación local de la tolerancia puede deducir las siguientes siete modificaciones:

$$P_1 := [120,320] \quad P_2 := [160,360] \quad P_3 := [0,200]$$
$$AP_2 := [0,40] \quad AP_3 := [0,40] \quad A := [120,160]$$
$$TOTAL := [280,480]$$

En este ejemplo, sólo se ha cambiado el límite de un parámetro en una red de restricciones simple. En general, los límites inferior y superior de varias variables pueden ser modificados de forma simultánea, y se pueden aparecer restricciones más complejas que simple sumas.

La propagación local de la tolerancia refina problemas de satisfacción de restricciones intervalares monótonamente inconsistentes pero admisibles. El algoritmo de propagación de la tolerancia se ha extendido también para tratar problemas de satisfacción de restricciones intervalares inadmisibles. En este caso las tolerancias proporcionan el mecanismo para determinar el mínimo ensanchamiento de los intervalos para las variables inadmisibles basado en la información presente en el problema de satisfacción de restricciones intervalares. Este tipo de paso de inferencia no monotónico es necesario de cara a ayudar al usuario de una hoja de cálculo basada en intervalos en la reformulación de problemas de satisfacción de restricciones intervalares de forma inteligente.

10.5 Las Condiciones de Aplicación

- **El problema**

En los algoritmos de propagación de la tolerancia local y de filtrado de Waltz, las variables intervalares deben ser calculadas respecto a las variables relacionadas de una u otra forma. Sin embargo, dichos

cálculos no siempre se pueden realizar directamente. Las precondiciones para los intervalos de los problemas de restricciones intervalares debe ser establecidas por dos razones fundamentales:

(a) **Definibilidad de las restricciones**. Ciertos valores para las variables deben ser excluidos de la definición de las restricciones. Por ejemplo, en la restricción X/Y=Z todas las situaciones $0 \in Y$ son implícitamente no permitidas. En general, Y puede ser una expresión de complejidad arbitraria.
(b) **Aplicabilidad de las funciones intervalares**. Las funciones intervalares necesarias no siempre se pueden definir o aplicar fácilmente:

◊ *Definibilidad.* Una función solución puede no ser definida para todos los valores de los argumentos. Por ejemplo, en la restricción $X*Y=Z$, la función solución $X=Z/Y$ no se puede calcular si $0 \in Y$. Sin embargo, en este caso, el valor $Y:=0$ no es excluido por la definición de la restricción tal como sucede con la restricción $X*Y=Z$: dicha restricción puede ser satisfecha por cualquier X si $Y:=0$ y Z:=0.

◊ *Múltiples Valores Discontinuos.* Una función solución puede no evaluarse en un único intervalo. Por ejemplo, en la restricción $X^2=Y$, la función solución para X debe evaluar dos valores $X:=[1,2]$ y $X:=[-1,-2]$ si $Y:=[1,4]$.

◊ *Monotonicidad.* Para funciones intervalares simples, como las funciones racionales, los valores se obtienen fácilmente considerando las cotas inferior y superior de los argumentos intervalares porque las funciones se comportan monótonamente. Sin embargo, en general y para funciones más complicadas este no es el caso, precisando un análisis intervalar más complejo.

A continuación, presentaremos dos técnicas generales para resolver los problemas de aplicación (1) y (2) que acabamos de introducir. Ambas técnicas se basan en la idea de descomponer un problema de satisfacción de restricciones intervalares ICSP de forma exhaustiva en un conjunto de subsituaciones que se comporten bien sin perder o introducir nuevas soluciones. En primer lugar, vamos a mostrar como se puede conseguir dicha descomposición.

- **Descomposición de Situaciones**

Un conjunto de divisiones D_i para las variables P_i con i=1,...,n de un problema de satisfacción de restricciones intervalares define su **descomposición** en un conjunto S de subsituaciones, es decir, en un **espacio intervalar**:

$$S = \{\{P_1 := X_1', \cdots, P_n := X_n'\} | X_i' \in D_i, i = 1, \cdots, n\} \qquad (10.30)$$

Por ejemplo, el espacio intervalar correspondiente a las divisiones $X \in [-\infty, 0^- | 0, 1]$ y $Y \in [-1, 1^- | 1, \infty]$ es

$\{\{X:=[-\infty,0^-], Y:=[-1,1^-]\}, \{X:=[-\infty,0^-], Y:=[1,\infty]\}, \{X:=[0,1], Y=[-1,1^-]\}, \{X:=[0,1], Y:=[1,\infty]\}\}$

Estamos interesados en descomposiciones S que satisfagan las siguientes propiedades:

(a) *"Completitud"*. Todo valor de solución exacta contenida en los intervalos de la solución original es una solución de una situación en el espacio S, es decir, la descomposición no pierde o duplica soluciones.
(b) *"Consistencia"*. Todo valor de solución exacta de las situaciones en S es una solución de la situación original, es decir, la descomposición no introduce nuevas soluciones exactas.

El problema de las condiciones de aplicación, presentado anteriormente, se puede resolver descomponiendo la situación original del problema de propagación de restricciones intervalares de una forma completa y consistente en un **espacio de aplicación**, en el cual se pueda llevar a cabo el razonamiento intervalar. Dicho espacio se determina de la siguiente manera:

Algoritmo para la determinación del Espacio de Aplicación

Paso 1. Para cada tipo de restricción (por ejemplo, $X*Y=Z$, $X/Y=Z$, etc.), se determina un **espacio de definición local** en el cual se define la restricción. Supondremos que las divisiones de las variables son finitas.

Paso 2. Cada espacio de definición local (obtenido en el paso 1) se descompone posteriormente en el **espacio de aplicación local** de cara a hacer posible la definición de las funciones solución, de forma conveniente.

Paso 3. Para una determinada red de restricciones, los espacios locales de aplicación (obtenidos en el paso 2) se combinan en un **espacio de aplicación global**, en el cual es posible la propagación intervalar.

Paso 4. El espacio de aplicación real para un determinado problema de satisfacción de restricciones intervalares se construye mediante la intersección del espacio de aplicación global (obtenido en el paso 3) con los intervalos del problema de satisfacción intervalar.

Los pasos 1-3 se pueden realizar "*a priori*" para una determinada red de restricción de un problema de satisfacción de restricciones intervalares. Sólo el paso 4 se debe realizar dinámicamente con respecto a una determinada situación. A continuación, vamos a describir con más detalles cada uno de los pasos anteriores.

A. Descripción del Paso 1: "Espacio Local de Definición"

Cada función restricción primitiva ($X/Y=Z$, $\log_X Y=Z$, etc.) define un espacio de definición local dentro del cual se define la restricción y que contiene todos los valores exactos posibles de la restricción. Por ejemplo, el espacio de definición local de la restricción $\log_X Y=Z$ tiene dos situaciones:

$$\{\{X\in[0^+,1^-]\cup[1^+,\infty],\ Y:=[0^+,\infty],\ Z:=[-\infty,\infty]\}\}$$

puesto que la base X de la función logaritmo no puede ser negativa o igual a 1 y el argumento Y debe ser positivo. Mientras que el espacio definición local de la restricción $Y=X^Z$ sería diferente.

B. Descripción del Paso 2: "Espacio Local de Aplicación"

De cara a garantizar que las funciones solución se puedan definir y calcular fácilmente, a menudo, se debe descomponer el espacio de definición local en el espacio de aplicación local. A continuación, se presentan dos criterios útiles para esta tarea:

◊ *Criterio de Definibilidad.*

En primer lugar, una función solución $X_1=F(X_2,\cdots,X_n)$ debe estar definida para todos los valores de los argumentos $x_2\in X_2,\cdots,x_n\in X_n$ y debe evaluar un intervalo único continuo dentro de X_1 (de otra manera F no sería una función). Por ejemplo, el espacio de definición de la restricción multiplicación $X*Y=Z$ es

$$\{\{X:=[-\infty,\infty],\ Y:=[-\infty,\infty],\ Z:=[-\infty,\infty]\}\}$$

Sin embargo, la función solución $Y=Z/X$ en $X*Y=Z$ no se puede calcular si $0\in X$ aunque $X:=0$ es un valor posible en el espacio de definición. Una situación parecida se produce para la función $X=Z/Y$. Descomponiendo aún más el espacio de definición se resuelve el problema: casos excepcionales con $0\in X$ y $0\in Y$ se pueden considerar separadamente utilizando el siguiente espacio de aplicación local de nueve situaciones:

$$\{\{X\in[-\infty,0^-]\cup[0^+,\infty],\ Y\in[-\infty,0^-]\cup[0^+,\infty],\ Z:=[-\infty,\infty]\}\}$$

Ahora las situaciones $0 \in X$ o $0 \in Y$ sólo se producen cuando $X:=0$, o bien, $Y:=0$, y las funciones solución para X e Y en estos casos puedan definirse a partir de las restricciones $0*Y=Z$ y $X*0=Z$, respectivamente:

$$X: \quad \text{si } Y=0 \text{ entonces } X \text{ sino } Z/Y$$

$$Y: \quad \text{si } X=0 \text{ entonces } Y \text{ sino } Z/X$$

$$Z: \quad X*Y$$

Intuitivamente, en el espacio de aplicación local, los casos excepcionales se separan en subespacios distintos para los cuales las funciones solución se pueden definir separadamente mediante las diferentes ramas de las definiciones de función condicionales.

◊ *Criterio de Monotonicidad.*

Al evaluar una función solución intervalar, los puntos máximo y mínimo de la correspondiente función exacta deben ser determinados. Esto puede ser realizado de forma conveniente mediante la descomposición del espacio de definición en subsituaciones en las cuáles las funciones solución puedan ser definidas en términos de las cotas de los argumentos intervalares, de la misma forma que para las operaciones básicas. Una condición suficiente para poderlo realizar es la monotonicidad: diremos que una función intervalar $P_1 = F(P_2, \cdots, P_n)$ es **monótona** en una situación $\{P_1 := X_1, \cdots, P_n := X_n\}$ si y sólo si la correspondiente función exacta f es monótona con respecto a cada argumento P_i dentro del intervalo X_i para i=2,...,n. Esto significa que si incrementamos cualquier argumento dentro de su intervalo, el valor de la función siempre, o bien, se incrementa, o bien, se decrementa independientemente de los valores de los otros argumentos (dentro de sus intervalos). Por ejemplo, $Z = X/Y$ es monótona cuando $X \subseteq [0,\infty]$ y $Y \subseteq [0^+,\infty]$, porque $z = x/y$ se incrementa al aumentar $x, y \in [0^+,\infty]$, y se decrementa al aumentar y, $x \in [0,\infty]$. Las condiciones de monotonicidad se pueden determinar considerando las derivadas parciales $df(x_1,\cdots,x_n)/dx_i$ para i = 1, ..., n de la función: la función $Y = F(X_1,\cdots,X_n)$ es monótona si las derivadas parciales no cambian de signo dentro de los intervalos dados $x_i \in X_i$ para i = 1, ... , n.

Una función monótona necesariamente tiene sus valores mínimo y máximo en los extremos del los intervalos dados. Por lo tanto, el valor de una función solución monótona $Y = F(X_1,\cdots,X_n)$ correspondiente al valor de una función exacta $y = f(x_1,\cdots,x_n)$ se puede calcular fácilmente mediante

$$Y = [\min(\{f(\overline{x}_1,\cdots,\overline{x}_n) | \overline{x}_i = \min(X_i), \text{ o bien, } \overline{x}_i = \max(X_i) \text{ para } i = 1,\cdots,n\}),$$
$$\max(\{f(\overline{x}_1,\cdots,\overline{x}_n) | \overline{x}_i = \min(X_i), \text{ o bien, } \overline{x}_i = \max(X_i) \text{ para } i = 1,\ldots,n\})]$$

donde $\min(\{v_1,\cdots,v_n\})$ y $\max(\{v_1,\cdots,v_n\})$ significan los valores máximo y mínimo v_i para i = 1,...,n, respectivamente, y min(X) y max(X) son los mínimos y máximos del intervalo X.

La monotonicidad garantiza que las cotas de las función solución se puedan calcular a partir de las cotas de los argumentos. Sin embargo, una función puede tener sus valores máximos y mínimos en los extremos de los intervalos de sus argumentos pero ser no monótona. En este caso, la monotonicidad es un criterio innecesariamente fuerte y el espacio de aplicación consistirá de muchas situaciones innecesarias. En principio, esto no es un problema puesto que no se pierde o se introduce erróneamente ninguna solución puesto que la descomposición es completa y consistente. Sin embargo, una descomposición innecesariamente grande conlleva cálculos innecesarios.

Por ejemplo, la restricción $X^a = Y$ con $a \in \{2,4,\cdots\}$ tiene el espacio de definición $\{\{X:=[-\infty,+\infty], Y:=[0,+\infty]\}\}$ y las funciones solución exactas

$$x = f_x(y) = \pm y^{1/a}$$

$$y = f_y(x) = x^a$$

La función intervalar para X está definida, es decir, se evalúa en un único intervalo: si los valores de X positivos y negativos se consideran por separado: $X \in [-\infty,0^{-}]\cup[0,\infty]$.
Con esta condición, las derivadas parciales

$$df_x(y)/dy = \pm(1/a)y^{1/a-1}$$

$$df_y(x)/dx = ax^{a-1}$$

no cambian de signo. El espacio de aplicación local es, por lo tanto

$$\{\{X \in [-\infty,0^{-}]\cup[0,\infty],\ Y:=[0,\infty]\}\}$$

Así para $X:=[x_1,x_2]$ e $Y:=[y_1,y_2]$ obtenemos las funciones solución siguientes

$$X:\quad \text{si } X \subseteq [0,\infty] \text{ entoces } [y_1^{1/a}, y_2^{1/a}] \text{ sino } [-y_2^{1/a}, -y_1^{1/a}]$$

$$Y:\quad [\min(x_1^a, x_2^a), \max(x_1^a, x_2^a)]$$

Estas definiciones resuelven el problema de los valores en múltiples intervalos. Por ejemplo, si $X:=[-\infty,\infty]$, $Y:=[1,4]$ y a=2, el problema puede ser considerado respecto a los valores $X:=[-\infty,0^{-}]$ y $X:=[0,\infty]$ separadamente, determinándose los dos resultados siguientes $X:=[-2,-1]$ y $X:=[1,2]$ para cada subsituación.
Por lo que hemos visto, la condición de monotonicidad sugiere que los valores de X positivos y negativos deben ser considerados separadamente. Esta es una condición estricta innecesariamente para situaciones donde Y es de la forma $Y:=[0,y_2]$ puesto que entonces los valores de X se podrían representar mediante un intervalo continuo. Por ejemplo, si $Y:=[0,4]$ y a = 2, entonces $X:=[-2,2]$. Mientras que si utilizaremos la descomposición propuesta anteriormente obtendríamos dos soluciones $X:=[-2,0^{-}]$ y $X:=[0,2]$. La solución más general $X:=[-2,2]$ se podría obtener generalizando las dos soluciones mediante la operación $\cup_s$ sobre la retícula de situaciones. Sin embargo, sería más eficiente no considerar el intervalo X en partes sino definir la función solución directamente de tal forma que el caso especial $Y:=[0,y_2]$ fuera evaluada correctamente.
Situaciones más complicadas se producen si al exponente a anterior se le permite tomar valores intervalares, es decir, es una variable. Hagamos que la restricción resultante $X^Y = Z$ tenga el espacio de definición

$$\{\{X:=[0,\infty],\ Y:=[-\infty,\infty],\ Z:=[0,\infty]\}\}$$

Las funciones solución exactas son:

$$x = f_x = z^{1/y}$$

$$y = f_y = \log_x(z)$$

$$z = f_z = x^y$$

Las condiciones de definibilidad para estas funciones nos proporcionan las siguientes divisiones:

$$X \in [0\,0^{+}, 1\,1^{+}, \infty] \quad \text{debido a } y = \log_x(z)$$

$$Y \in [-\infty,0^{-}\,0\,0^{+},\infty] \quad \text{debido a } z = x^{1/y}$$

$$Z \in [0\,0^{+},\infty] \quad \text{debido a } y = \log_x(z)$$

Las derivadas parciales de las funciones solución son:

$$df_x/dz = z(1/y-1)/y, \qquad df_x/dy = (z^{1/y})(\ln(z))(-y^{-2})$$

$$df_y/dz = 1/(z\ln(x)), \qquad df_y/dx = -\ln(z)/(x\ln(x)^2)$$

$$df_z/dy = x^y \ln(x) \qquad df_z/dx = yx^{y-1}$$

A partir de las derivadas df_x/dy y df_y/dx se puede establecer un punto adicional de división., Z=1, debido al criterio de monotonicidad. Utilizando la división refinada $Z \in [0|0^+, 1^-|1, \infty]$, se obtienen las siguientes funciones solución

$X = [x_1, x_2]$ si $Y = 0$ entonces (si $Z = 0$ entonces 0 sino X),
sino si $Z = 0$ entonces 0,
sino $[\min(\{z_i^{1/y_i}\}), \max(\{z_i^{1/y_i}\})]$ para $i = 1,2$;

$Y = [y_1, y_2]$ si $Z = 0$ entonces (si $X = 0$ entonces Y sino $\{\}$)
sino si $X = 1$ entonces Y,
sino $[\min(\{\log_{x_i}(z_i)\}), \max(\{\log_{x_i}(z_i)\})]$ para $i = 1,2$;

$Z = [z_1, z_2]$ si $X = 0$ entonces 0
sino $[\min(\{x_i^{y_i}\}), \max(\{x_i^{y_i}\})]$ para $i = 1,2$.

Por ejemplo, ningún valor (lo cual se ha indicado mediante $\{\}$) para Y puede satisfacer la restricción $X^Y = Z$ si $Z = 0$ y $X \neq 0$.

Como conclusión, las funciones solución locales y el espacio de aplicación local se determinan mediante la división del espacio local de definición de cada restricción de forma que las funciones solución se puedan definir de forma conveniente en cada subsituación. Los criterios de definibilidad y monotonicidad proporcionan las reglas para seleccionar los puntos de división: las funciones solución deben estar definidas y ser monótonas en cada subsituación.

C. Descripción del Paso 3: "Espacio Global de Aplicación"

Una variable X se puede utilizar en una red de restricciones en k restricciones, $k \geq 1$, que restringen los valores de X con divisiones de aplicación locales $D_1, \ldots, D_k$. Diremos que **la división de aplicación global** X es

$$D = \{X_1 \cap \cdots \cap X_k \mid X_i \in D_i, i = 1, \cdots, k\} \qquad (10.31)$$

Esta división establece de forma intuitiva los rangos en los cuales todas las variables en la red esta definidas con respecto a X y las funciones solución en las cuales X se puede utilizar como argumento. Ninguna valor exacto de X es excluido o introducido en las posibles soluciones. Esto significa que la aplicabilidad de la propagación intervalar en una red de variables $P_1,\ldots,P_n$ con respecto a todas las restricciones y todas las funciones solución se puede garantizar de una forma completa y consistente en el espació de aplicación global S definido como

$$S = \{\{P_1 := X_1, \cdots, P_n := X_n\} \mid X_i \in D_i \text{ es la división global de aplicación de } P_i, i = 1, \cdots, n\}$$

Por ejemplo, consideremos la red de restricciones

$$X^2 + PX = Q \equiv \{X^2 = Y_1, P * X = Y_2, Y_1 + Y_2 = Q\}$$

Los espacios locales de aplicación para las restricciones $P*X=Y_2$ y $X^2=Y_1$ se han presentado en el apartado anterior, mientras que el espacio local de aplicación para la restricción suma $X+Y=Z$ es $\{\{X:=[-\infty,\infty], Y:=[-\infty,\infty], Z:=[-\infty,\infty]\}\}$. El espacio global de aplicación de la restricción $X^2+PX=Q$ es, por lo tanto

$$S=\{\{X\in[-\infty,0^-]0,\infty]\cap[-\infty,0^-]0^+,+\infty]=[-\infty,0^-]0^+,+\infty],$$
$$Y_1:=[0,\infty]\cap[-\infty,\infty]=[0,\infty],$$
$$P\in[-\infty,0^-]0^+,+\infty],$$
$$Y_2:=[-\infty,\infty]\cap[-\infty,\infty]=[-\infty,\infty],$$
$$Q:=[-\infty,\infty]\}\}$$

La notación $X\in D$, aquí significa que X puede ser cualquier intervalo constituyente dentro de la división D. En este caso S representa un conjunto de nueve situaciones intervalares.

D. Descripción del Paso 4: "Espacio Real de Aplicación"

Supongamos que un problema de satisfacción de restricciones intervalares ICSP con los siguientes intervalos iniciales para las variables del problema $S=\{P_1:=X_1,\cdots,P_n:=X_n\}$ y con el espacio de aplicación global $G=\{\{P_1:=Y_1\in D_1,\ldots,P_n:=Y_n\in D_n\}\}$. De cara a resolver el problema de satisfacción de restricciones intervalares ICSP tendremos que considerar S con respecto a cada subsituación en G, es decir, con respecto a las situaciones:

$$\begin{aligned}S'=&\{\{P_1:=X_1\cap Y_1,\cdots,P_n:=X_n\cap Y_n\}|\\&\{P_1:=Y_1,\cdots,P_n:=Y_n\}\in G\}\\=&\{\{P_1\in X_1\cap D_1,\cdots,P_n\in X_n\cap D_n\}\}\end{aligned}\qquad(10.32)$$

S' se denomina el **espacio de aplicación actual**. El espacio de aplicación actual contiene todas las soluciones exactas posibles del problema ICSP en exactamente una de sus subsituaciones.

Por ejemplo, el problema de resolver la ecuación $X^2-2X=3$ corresponde a un problema de satisfacción de restricciones intervalares con la red

$$X^2+PX=Q\equiv\{X^2=Y_1, P*X=Y_2, Y_1+Y_2=Q\}$$

y con la situación inicial

$$\{X:=[-\infty,\infty], P:=-2, Q:=3, Y_1:=[-\infty,\infty], Y_2:=[-\infty,\infty]\}$$

El espacio de aplicación actual se obtiene por intersección de la situación inicial con el espacio de aplicación global.

$$\{\{X\in[-\infty,0^-]0^+,+\infty], Y_1\in[0,\infty],$$
$$P:=-2, Y_2:=[-\infty,+\infty], Q:=3\}\}$$

- **Búsqueda en el espacio de aplicación**

Un problema de satisfacción de restricciones intervalares puede ser resuelto después de determinar su espacio de aplicación actual S. Cada subsituación de S puede contener soluciones. El algoritmo que se presenta a continuación realiza una búsqueda de las soluciones de forma exhaustiva para un problema de satisfacción de restricciones intervalares sobre el espacio de aplicación actual. La idea de dicho algoritmo consiste en instanciar las variables en un orden determinado sobre los constituyentes de los intervalos de las divisiones que definen el espacio de aplicación actual. Después de cada instanciación, todas las funciones solución aplicables y aún no satisfechas respecto a las variables ya instanciadas son

aplicadas. De esta forma, los extremos de los intervalos no utilizados se detectan rápidamente y se comparten los cálculos comunes en diferentes subsituaciones. Una buena heurística para escoger el orden de instanciación es instanciar los nodos con divisiones pequeñas, los intervalos estrechos, y con elevada conectividad, en primer lugar. De esta forma, los extremos no utilizados es probable que se encuentren más fácilmente y que los cálculos más comunes sean compartidos.

Algoritmo de Resolución de un problema ICSP

Paso 1. Soluciones: ={}
Paso 2. Divs := divisiones que definen el espacio de aplicación actual del problema de satisfacción de restricciones intervalares ICSP en un determinado orden.
Paso 3. Si alguna D ∈ Divs está vacía, entonces devolver {}
Paso 4. Instanciar las divisiones (Divs:=Divs, Vars:={}, S:={})
Paso 5. Devolver $\cup_s$ Soluciones.

Vamos a continuación a detallar el procedimiento para instanciar las divisiones.

Algoritmo de Instanciación de Divisiones (Divs, Vars, S)

Paso 1. Si Divs = {}, entonces Soluciones := {S} ∪ Soluciones, y volvemos.
Paso 2. D := la siguiente división de variable in Divs.
Paso 3. Var := la variable de D
Paso 4. Vars := {Var} ∪ Vars (Variables Instanciadas)
Paso 5. Fns := $\{X_1 = F(X_2, ..., X_n) \mid X_i \in Vars, i=1,...,n\}$ (Funciones Solución aplicables del problema de satisfacción de restricciones intervalares con las variables instanciadas)
Paso 6. NewFns := $\{X_1 = F(X_2,...,X_n) \in Fns \mid Var \in \{X_1, ..., X_n\}\}$ (Estas Funciones Solución son aplicables debido a Var)
Paso 7. Para cada Int ∈ D realizar: (Búsqueda sobre las divisiones constituyentes)
S' := S ∪ {Var := Int} (Instanciación de Var en la situación actual S)
S':= aplicar el Algoritmo de propagación local de la tolerancia con Agenda := NewFns y situación inicial S', pero sólo considerando funciones aplicables F∈ Fns durante la propagación.

Si S' ≠ {}, entonces instanciar las divisiones (Vars: = Vars, S:= S', Divs: = Divs - D) (Recursivamente) sino continuar con el siguiente Int en el paso (7) (para determinar los extremos inadmisibles)

Por ejemplo, al resolver

$$X^2 + PX = Q \equiv \{X^2 = Y_1, P*X = Y_2, Y_1 + Y_2 = Q\}$$

en el espacio actual de aplicación

$$\{\{X \in [-\infty, 0^-|0|0^+, +\infty], Y_1 \in [0, \infty],$$
$$P:=-2, Y_2:=[-\infty,+\infty], Q:=3\}\}$$

podemos utilizar el siguiente orden P, Q, Y_1, Y_2, X. Después de instanciar P:=−2, Q:=3, Y_1:=[0,∞] y Y_2:=[−∞,∞], las funciones solución de la restricción $Y_1 + Y_2 = Q$ pueden ser aplicadas mediante el procedimiento de propagación local. La situación resultante se considera mediante la instanciación de X con los constituyentes de $[-\infty, 0^-|0|0^+, \infty]$ y mediante funciones solución adicionales (*NewFns*) que tienen X como un argumento (funciones originadas a partir de $X^2 = Y_1$ y $P*X = Y_2$). Como resultado, encontraremos mediante el procedimiento de propagación local las dos raíces X:=−1 y X:=3 como

puntos fijos correspondientes a las instanciaciones $X:=[-\infty,0^-]$ y $X:=[0^+,\infty]$, respectivamente ($X:=0$ se encuentra que es inadmisible.

- **Propagación de las Divisiones**

Aunque el algoritmo que hemos presentado para resolver problemas de satisfacción de restricciones intervalares funciona bien en este último ejemplo, su aplicación en general presenta, a menudo, problemas. Es computacionalmente costoso porque los cálculos a lo largo de diferentes caminos de búsqueda pueden ser en parte redundantes (las funciones solución deben ser evaluadas en situaciones que se solapan). Además, se puede acabar generando muchas soluciones aunque originalmente nosotros sólo estuviéramos interesados en determinar un único intervalo (o división) solución. En la práctica, el algoritmo que hemos presentado sólo puede ser aplicado a problemas pequeños.

Para resolver los problemas que presenta este algoritmo, se propone la generalización de la propagación intervalar a una **propagación de divisiones**. En este esquema se pueden propagar inmediatamente valores en el espacio de aplicación con todas las variables instanciadas en divisiones, obteniéndose una solución constituida por una única división. Aunque el cálculo con divisiones es ligeramente más complicado que con los intervalos, la propagación de divisiones, tal como se describirá a continuación , es más adecuada en situaciones que presenten grandes espacios de aplicación (es decir, espacios consistentes de varias subsituaciones intervalares).
Puede parecer que en la propagación de divisiones todo lo que se debe hacer es generalizar las definiciones de funciones solución intervalares en funciones solución sobre divisiones. Sin embargo, aparece que la aplicación de tales funciones en muchos casos subdivide una división en un conjunto de intervalos cada vez mayor. y la propagación cada vez se convierte en computacionalmente ineficiente. Por ejemplo, supongamos que $X:=[-\infty,0^-|0^+,\infty]$. Si una de sus funciones solución evalúa el intervalo $[-\infty,1^-|1|1^+,10]$, entonces se generará un nuevo punto de división $X:=[-\infty,0^-|0^+,1^-|1|1^+,10]$. En problemas que convergen asintóticamente, el número de intervalos de una división puede crecer mucho. Por ejemplo, esto le sucede a X en $X^2 + PX = Q$ en la situación

$$\{\{X \in [-\infty,0^-|0|0^+,+\infty], Y_1 \in [0,\infty],$$
$$P:=-2, Y_2:=[-\infty,+\infty], Q:=3\}\}$$

El fenómeno de crecimiento de las divisiones en la propagación de las mismas se denomina el **problema de subdivisión** ("splitting problem"). A continuación, mostraremos en primer lugar como definir las funciones solución sobre divisiones y después como se puede resolver el problema de subdivisión.

A. Aritmética sobre Divisiones

Si se representan las funciones sobre intervalos y sobre divisiones correspondientes a la función exacta f por f_i y f_d, respectivamente (por ejemplo, $+_i$ significa suma intervalar). Las funciones división $f_d(D_1,\ldots,D_n)$ correspondientes a una función intervalar $f_i(I_1,\ldots,I_n)$ se puede definir como sigue:

$$f_d(D_1,\cdots,D_n) = \cup\{f_i(I_1,\cdots,I_n) | I_1 \in D_1,\cdots,I_n \in D_n\} \qquad (10.33)$$

Intuitivamente, la función intervalar se aplica a cada combinación de los intervalos constituyentes de los argumentos, lo cual garantiza que los intervalos resultantes contienen exactamente todos los posibles valores de la función exacta f. El operador unión crea la correspondiente división a los intervalos que pueden presentar posiblemente partes solapadas. Por ejemplo, generalizando la multiplicación intervalar $*_i$ definida anteriormente a la multiplicación de divisiones $*_d$, el siguiente ejemplo se evaluaría de la siguiente manera

$$[-2,1]\![3,4]*_d [2,3]\![4,5] =$$
$$([-2,-1]*_i [2,3]) \cup ([-2,-1]*_i [4,5])$$
$$\cup([3,4]*_i [2,3]) \cup ([3,4]*_i [4,5]) =$$
$$[-6,-2] \cup [-10,-4] \cup [6,12] \cup [12,20] =$$
$$= [-10,-2]\![6,20]$$

El número de constituyentes en la división resultante varía entre 1 y $k_1 * k_2 * \cdots * k_n$, donde k_i es el número de intervalos de la división D_i, i = 1,...,n. En el ejemplo anterior, el máximo número 2*2 = 4 se redujo a 2 porque los dos pares de intervalos se solapaban y podían combinarse mediante el operador unión.
Cuando se generaliza la propagación intervalar a la propagación de divisiones no deben realizarse suposiciones adicionales respecto a las condiciones de aplicación porque la definición de la evaluación de una función sobre divisiones se reduce a la evaluación de dicha función sobre intervalos. El espacio de aplicación real, por lo tanto, puede ser utilizado también para la propagación de divisiones.

B. Solución al Problema de las Subdivisiones

Como una solución al problema de las subdivisiones, se propone la aplicación del siguiente **principio de "no subdivisión"**: en la propagación de divisiones presentada en el algoritmo de propagación de la tolerancia la operación de intersección del paso (3.2) se reemplaza por el operador de intersección de no subdivisión de divisiones $\cap_n$

$$D' \cap_n D = \cup\{[\min(X_i \cap_n D'), \max(X_i \cap_n D')] | X_i \in D\} \qquad (10.34)$$

donde D' es el valor de una nueva división evaluada mediante una función solución y D es el valor de la antigua división en la situación S. Intuitivamente, los constituyentes de una división se puede reducir sólo desde ambos extremos pero no subdividir más. A diferencia de resolución de problemas de propagación intervalar ICSP, el algoritmo de propagación de la tolerancia modificado a fin de que utilice el principio de no subdivisión no considera los subespacios del problema ICSP separadamente sino que determina la división solución directamente.
El principio de no subdivisión hace que la propagación de las divisiones sea computacionalmente factible. Sin embargo, el precio que se paga es que, en general, se cambia la interpretación de los intervalos en la solución. No se puede ya garantizar que cualquier valor en los intervalos solución sea consistente (ya sea global o localmente), es decir, sean extensibles a una solución completamente exacta (local o globalmente). Sin embargo, se puede asegurar al menos que las cotas inferior y superior de las divisiones constituyentes se puedan extender a soluciones completamente exactas (local o globalmente) y que todas las otras soluciones deben residir dentro de las divisiones. Por ejemplo, para el problema ICSP

$$X + Y = Z$$
$$X := [1,2]\![8,9]$$
$$Y := 0$$
$$Z := [-\infty, \infty]$$

obtenemos

$$Z = X +_d Y = ([1,2] + [0,0]) \cup ([8,9] + [0,0]) = [1,2]\![8,9]$$

(utilizando la suma intervalar definida anteriormente). Sin embargo, aplicando el principio de no división obtenemos

$$Z := [1,2]\![8,9] \cap_n [-\infty, \infty] = [1,9]$$

aunque los valores $2 < Z < 8$ no sean consistentes. Utilizando la propagación de divisiones junto con el principio de no división no se pueden filtrar los valores dentro de un intervalo constituyente de una

división si los valores de los extremos son consistentes. La propagación de divisiones es inferencialmente más débil que las propagación de intervalos puros combinados con el procedimiento de búsqueda presentado anteriormente para la resolución de problemas ICSP. Sin embargo, la propagación de divisiones nunca pierde o introduce nuevas soluciones. Por ejemplo, si aplicamos la propagación de divisiones junto al principio de no división a

$$X^2 + PX = Q \equiv \{X^2 = Y_1, P * X = Y_2, Y_1 + Y_2 = Q\}$$

con el espacio de aplicación real

$$\{\{X \in [-\infty, 0^-|0|0^+, +\infty], Y_1 \in [0, \infty],$$

$$P := -2, Y_2 := [-\infty, +\infty], Q := 3\}\}$$

no podremos determinar las soluciones exactas X := -1 y X := 3 que se habían obtenido anteriormente mediante el algoritmo de solución de problemas ICSP. En su lugar, obtendremos la división solución $X := [-1.5, \cdots, 0^-|0|0^+, 3.0\ldots]$.

El principal beneficio que se obtiene de la propagación de divisiones es la posibilidad de tratar con espacios de aplicación grandes en los cuales el algoritmo que habíamos presentado anteriormente se volvía computacionalmente inviable. Después de determinar las cotas aproximadas de los valores posibles mediante propagación de divisiones se pueden obtener soluciones más precisas considerando los diferentes intervalos constituyentes separadamente. Si se utilizan sólo divisiones continuas, entonces la propagación de divisiones es equivalente a la propagación de intervalos, mientras que si los intervalos son simples valores para las variables conocidas ("entradas") y intervalos grandes para las variables desconocidas ("salidas"), entonces la propagación de divisiones es idéntica a la propagación de valores exactos.

10.6 Consistencia Global

- **Consistencia Local versus Global**

La consistencia global siempre implica la consistencia local; cualquier solución global G es una especialización de la local LGCS L, $G \subseteq_s L$. La propagación de la tolerancia local nunca pierde o introduce nuevas soluciones globales. Sin embargo, presenta dos grandes problemas:

(a) Determinación de la solución más ajustada

Mediante técnicas de consistencia local se pueden determinar cotas exteriores seguras para la solución aunque no necesariamente las más ajustadas. Consideremos el problema ICSP siguiente:

$$X + T = Y$$

$$Y + T = Z$$

$$X := 1$$

$$T := [0, \infty]$$

$$Y := [0, \infty]$$

$$Z := 11$$

Utilizando propagación local obtendremos la solución local:

$$\{X := 1, T := [0,10], Y := [1,11], Z := 11\}$$

Esta solución no es globalmente consistente porque T e Y no son globalmente consistentes. Si, por ejemplo, T = 9 obtendremos Y = X + T = 10 de la primera restricción, sin embargo, la segunda restricción no puede ser satisfecha debido a que $Y + T = 19 \notin Z = [11,11]$. La LGCS global del problema ICSP es la siguiente

$$\{X := 1, T := 5, Y := 6, Z := 11\}$$

(b) Detección de inconsistencia global

Si un problema ICSP no tiene solución global, ello no implica que no pueda tener una solución local. Por ejemplo, la solución local del polinomio

$$X^4+2X^3+3X^2+4X+5=0$$

compuesto de las restricciones $X^n=Y$, $X*Y=Z$ y $X+Y=Z$ es

$$X:=[-2.34\ldots,-0.1\ldots]$$

aunque el polinomio no tiene raíces. Lo que nos dice la solución local LGCS sólo es que si el problema tuviera soluciones globales estas deberían encontrarse dentro de las tolerancias de la solución local. Por lo tanto, si la propagación local no obtiene ninguna solución, entonces podemos afirmar con toda certeza que el problema no presenta ninguna solución ya sea exacta o global. Sin embargo, de la existencia de una solución local, no se puede deducir una solución global: la consistencia global implica la consistencia local, pero no viceversa.

Existen, sin embargo, categorías de problemas ICSP para los cuales la consistencia local implica la global, es decir, es equivalente a la consistencia global. Para tales problemas la propagación de la tolerancia local es suficiente para determinar las soluciones globales. En particular, este el caso de los problemas ICSP cuya red de restricciones es acíclica. A continuación presentamos un Teorema que se deriva directamente de la investigación sobre problemas de satisfacción de restricciones (CSP) discretos donde un problema CSP estructurado mediante un árbol consistente globalmente permite la generación de soluciones exactas:

***Teorema 10.1**: "Una red de restricciones acíclica es globalmente consistente si y sólo si es localmente consistente.*

La prueba de dicho Teorema, se puede realizar de forma intuitiva: seleccionando un valor para una variable los valores de la variable relacionados localmente se pueden escoger de forma adecuada en una situación localmente consistente si las variables no están relacionadas entre sí, es decir, la red es acíclica. Esto significa que las variables son globalmente consistentes, es decir, la consistencia local implica la consistencia global. Por otro lado, es evidente que la consistencia global implica la consistencia local.

Así por ejemplo, los problemas que aparecen en el problema ICSP

$$X+T=Y$$
$$Y+T=Z$$
$$X:=1,\quad T:=[0,\infty]$$
$$Y:=[0,\infty],\quad Z:=11$$

son debidos a la ciclicidad. En algunos casos, los ciclos se pueden eliminar reescribiendo partes de la ecuación de una forma acíclica equivalente. Por ejemplo, el problema ISCP anterior puede reescribirse de forma acíclica $Y=(X+Z)/2$ mediante la eliminación de T. Aplicando la propagación de la tolerancia local a esta red se obtendrá siempre la solución global.

A continuación presentamos otro Teorema que establece un caso especial importante en cual cualquier ICSP acíclico puede resolverse de forma global mediante propagación de la tolerancia local.

***Teorema 10.2**: "Si las variables de cualquier "cutset" (mínimo conjunto de nodos que cortan todos los ciclos de una red) de una red de restricciones S toman un valor exacto, entonces S es globalmente consistente si y sólo si es localmente consistente.*

Para probar este Teorema, también de forma intuitiva: asignando a las variables de un "cutset" valores exactos las transformamos en constantes exactas, de esta forma se rompen los ciclos y conseguimos que la red acíclica en la cual las condiciones de consistencia local y global son equivalentes.

Desafortunadamente, los problemas ICSP no siempre se pueden representar de forma acíclica o tienen un "cutset" de valor exacto. Para tales casos, se precisan técnicas de consistencias global más potentes. A continuación, vamos a presentar dos de dichas técnicas: la subdivisión dinámica y la propagación de la tolerancia global.

- **Subdivisión dinámica**

Los Teoremas 10.1 y 10.2 sugieren que nos debemos centrar en la variables de "cutset" de las restricciones para conseguir la satisfacción global de las restricciones. Esta técnica ha sido también utilizada en la generación de soluciones exactas en problemas de satisfacción de restricciones discretos. El procedimiento de **subdivisión dinámica** ("dynamic splitting") utiliza esta idea. En primer lugar, esta técnica determina la solución local fácilmente disponible S. Después de esto, S se descompone en subsituaciones respecto a una variable de "cutset". Para a continuación determinar las situaciones locales de entre de las subsituaciones, descomponiéndose el resultado de nuevo, y así hasta que se alcanza una solución globalmente consistente. Un cutset adecuado (o varios) para un problema ICSP puede ser determinados mediante una fase de compilación separada antes de aplicar el procedimiento en diferentes situaciones. El procedimiento supone en el paso (4) que se han podido identificar las soluciones globales o que se dispone de algún otro criterio para finalizar la subdivisión: por ejemplo, el número de subdivisiones recursivas puede estar acotado. Un criterio parcial para identificar las soluciones globales viene dado por el Teorema 5.2

Algoritmo de Subdivisión Dinámica de un problema ICSP

Paso 1. S := Local LGCS obtenida mediante el algoritmo de propagación de la tolerancia local
Paso 2. Si S={}, abortamos el algoritmo devolviendo "ICSP inadmisible".
Paso 3. Si el ICSP es acíclico, entonces devolvemos S como la solución global.
Paso 4. Si S es globalmente consistente (es decir , las variables de cualquier cutset tienen valores exactos) o si se satisface cualquier otro criterio de finalización, entonces devolvemos S como la solución.
Paso 5. Escogemos una variable de cutset $P:=X$ de S mediante algún criterio (por ejemplo, seleccionamos la variable cuyo intervalo tenga la mayor anchura).
Paso 6. Subdividimos X de forma exhaustiva en intervalos $\{X_1,\cdots,X_n\}$ mediante algún criterio (por ejemplo, biseccionando X).
Paso 7.Creamos la descomposición $\{S_1,\cdots,S_2\}$ correspondiente a $P:=X_1,\cdots,P:=X_n$.
Paso 8. Aplicamos la subdivisión dinámica a cada $S_1,\cdots,S_n$.
Paso 9. Devolvemos la generalización $\cup_s\{S_i'\}$, donde S_i' son las soluciones que se han obtenido en el paso (8). De forma opcional, el conjunto $\{S_i'\}$ se puede devolver si estamos interesados en obtener los conjuntos solución.

La subdivisión dinámica funciona porque con valores de variables más acotados, las soluciones globales se pueden determinar mediante criterios locales de una forma más precisa. Por ejemplo, consideremos el problema de determinar las raíces positivas de

$$X^3-9X+4=0$$

A partir del valor inicial $X:=[0,+\infty]$ la propagación de la tolerancia local determina la solución local LGCS $X:=[0.45\ldots,2.74\ldots]$ en el paso (1). En el paso (5), la variable de cutset X se puede subdividir. Si biseccionamos X en $X:=[0.45\ldots,1.6^-]$ y $X:=[1.6,2.74\ldots]$ y resolvemos los correspondientes ICSP (paso (8)), determinamos las soluciones globales X:=0.45... y X:=2.74... sin tener que realizar más subdivisiones.

La subdivisión dinámica ofrece un enfoque para resolver el problema de la incompletitud de las técnicas de relajación de valores exactos numéricos. Con estas técnicas, así como con el método iterativo de Newton, no se pueden siempre determinar todas las soluciones o determinar si existe una solución. Mediante la propagación de la tolerancia local, es posible considerar problemas con espacios infinitos sin perder soluciones. Si la propagación de la tolerancia local falla, podemos estar seguros que no existe solución dentro de los intervalos. Por ejemplo, el polinomio de cuarto orden

$$X^4 + 2X^3 + 3X^2 + 4X + 5 = 0$$

se encuentra que es inadmisible biseccionando la solución local

$$X := [-2.34\ldots, -0.1\ldots]$$

sólo una vez. De esta forma, se puede probar numéricamente que el polinomio anterior no tiene solución. La propagación de la tolerancia se podría también aplicar en esquemas de relajación con valores exactos para determinar los rangos adecuados para los valores iniciales de las variables de prueba/error.

Sin embargo, el algoritmo de subdivisión dinámica presenta dos grandes problemas:

(a) *La Condición de Finalización.* El procedimiento supone que se pueden identificar las variables y situaciones globalmente consistentes (paso (4)). En general, es difícil y es preciso utilizar alguna condición de finalización ad hoc. El Teorema 10.2 establece que sólo una condición innecesariamente estricta para la consistencia global. Se precisan criterios generales en situaciones globalmente subrestringidas en las cuales las variables de cutset pueden tener valores de tolerancia no exactos y ser aún globalmente consistentes. De otra forma, se puede llegar a una subdivisión posiblemente infinita y sin utilidad cuando se trata de determinar solución cada vez más precisas con respecto a las variables de cutset.

(b) *Eficiencia.* Incluso si se pudieran identificar las soluciones globalmente consistentes y su número fuera finito (o se dispusiera de otras condiciones de terminación), la subdivisión puede generar un gran número de soluciones intermedias localmente consistentes que se pueden identificar como globalmente inadmisibles sólo después de realizar posteriores subdivisiones, lo cual puede resultar computacionalmente ineficiente.

El algoritmo de propagación global de la tolerancia que se presenta a continuación es un intento de resolver estos problemas.

10.7 Propagación de la Tolerancia Global

- **La Idea**

Supongamos que la red de restricciones S con variables $P_1,\ldots,P_n$. La función $P_i = F(P_1, \cdots, P_{i-1}, P_{i+1}, \cdots, P_n)$ para $2 \le i \le n$, se denomina la **función solución global** respecto a S si evalúa los valores de P_i cuando las variables argumento $P_1, \cdots, P_{i-1}, P_{i+1}, \cdots, P_n$ varían independientemente dentro de sus tolerancias. Por lo tanto, P_i es globalmente consistente en S si y sólo si $P_i \subseteq F(P_1, \cdots, P_{i-1}, P_{i+1}, \cdots, P_n)$. Obviamente, la utilización de funciones solución globales en lugar de las correspondientes locales durante la propagación de la tolerancia local, permitirá determinar soluciones globales con respecto a S. Sin embargo, esto es normalmente infactible: en primer lugar, la resolución de funciones solución globales algebraicamente es normalmente difícil o imposible. En segundo lugar, de esta forma un problema ICSP siempre debería ser resuelto globalmente incluso si fuera computacionalmente más económico utilizar técnicas locales en alguna parte de la red de restricciones. En tercer lugar, sería deseable resolver el problema ICSP en partes locales simples cuando fuera posible y utilizar técnicas globales más complicadas sólo cuando fuera necesario.

Una de las ideas clave de esta sección es que la observación de que es suficiente con determinar funciones solución globales sólo con respecto a algunas variables críticas y sólo con respecto a algunas

subredes de la red de restricciones original. De esta forma las soluciones globales se pueden determinar sin tener que sacrificar todos los beneficios que comporta el razonamiento local simple. A continuación se presentan cuales son las funciones solución a la que se deben utilizar durante la propagación de la tolerancia de cara a garantizar la consistencia global. La técnica de utilizar funciones solución locales y globales junto con el algoritmo de propagación de la tolerancia local para resolver problemas ICSP se denomina **algoritmo de propagación de la tolerancia global**.

El Teorema 10.1 nos indica que la consistencia local no es equivalente a la consistencia global debido a la presencia de ciclos. Un ciclo es un conjunto de variables conectada circularmente unas a otras mediante una cadena de restricciones diferentes mútuamente a las que denominaremos **restricciones cíclicas**. Por ejemplo, la red de la Fig. 10.7 tiene un ciclo {T, Y} con dos restricciones cíclicas.

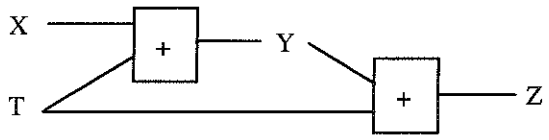


Fig. 10.7 Red de restricciones cíclica

Si todas las variables cíclicas fueran globalmente consistentes en una situación localmente consistente, entonces la situación sería también globalmente consistente. Además, es suficiente con determinar las funciones sólo con respecto a las restricciones cíclicas correspondientes a los ciclos formados por las variables, debido a que la consistencia local con respecto a las restricciones acíclicas restantes se alcanza mediante la propagación de la tolerancia local. Además, no se deben considerar ciclos que son subconjuntos de otros ciclos: es suficiente con tratar con el conjunto de los ciclos mayores LS que contengan todos los otros ciclos. Denominaremos una selección de ciclos de este tipo **loop cover**. De una forma más precisa, la agenda de funciones solución en el algoritmo de propagación de la tolerancia local que hace falta para la consistencia global en el algoritmo de propagación de la tolerancia global se obtiene mediante el procedimiento que presentamos a continuación.

Algoritmo para la determinación de la Agenda de la Propagación de la Tolerancia Global

Paso 1. Determinar las variables cíclicas $P_1, \cdots, P_n$ de la red de restricciones y su correspondiente loop cover LS. Sean los conjuntos de los mayores ciclos a través de $P_1, \cdots, P_n$: $LS_1 \subseteq LS, \cdots, LS_n \subseteq LS$, respectivamente.

Paso 2. Determinar el conjunto G_i de todas las funciones solución para cada variable P_i, i=1,...,n, correspondientes a los ciclos $L \in LS_i$. Las funciones solución para cada P_i, i=1,...,n, en la agenda son: $G_i \cup$ {Funciones Solución Locales para P_i excluyendo aquellas funciones locales que se originan a partir de las restricciones cíclicas correspondientes a los ciclos LS_i (es decir, los ciclos a través de P_i)}.

Paso 3. Para las variables no cíclicas $P_{j \neq i}$ añadir las funciones solución normales en la agenda.

- **Determinación de las funciones solución globales**

Las funciones solución globales de una variable P con respecto a un ciclo (paso (2) del algoritmo anterior) se pueden determinar en dos pasos:

Paso (A). Se construye una ecuación algebraica simple equivalente a las restricciones cíclicas del ciclo eliminando las otras variables cíclicas.

Paso (B). Obtenemos P de la ecuación obtenida en (A) algebraicamente, es decir, deducimos la función solución global.

El paso (B) puede ser difícil o imposible de realizar debido a las limitaciones del álgebra o del solver de problemas algebraicos utilizados. Además, la reglas algebraicas intervalares deben ser tenidas en cuenta. En primer lugar, se presentar un ejemplo en el cual estos problemas no son importantes para continuación discutir el caso general.

El problema ICSP siguiente

$$X+T=Y$$
$$Y+T=Z$$
$$X:=1, \quad T:=[0,\infty]$$
$$Y:=[0,\infty], \quad Z:=11$$

presenta, tal como hemos visto anteriormente, un ciclo $\{Y,T\}$. Las ecuaciones para resolver Y y T se obtienen eliminando T y Y respectivamente (paso A):

$$X+(Z-Y)=Y$$
$$X+T=Z-T$$

Las funciones solución globales para Y y T se obtienen fácilmente en el paso (B):

$$Y=(X+Z)/2$$
$$T=(Z-X)/2$$

La agenda para la propagación de la tolerancia global se obtienen mediante el algoritmo presentado anteriormente, de la siguiente manera:

Paso 1. Las variables del ciclo son $\{Y,T\}$ y los ciclos a través de ellas son $LS=\{\{Y,T\}\}$.
Paso 2. Las correspondientes funciones solución globales son:

$$Y=(X+Z)/2$$
$$T=(Z-X)/2$$

Las funciones solución locales:

$$Y=X+T$$
$$Y=Z-T$$
$$T=Y-X$$
$$T=Z-Y$$

ya no se utilizarán más debido a que son debidas a las restricciones cíclicas.
Paso 3. Para las otras variables X y Z se utilizan las funciones solución locales $X=Y-T$ y $Z=Y+Z$.
La agenda para la propagación de la tolerancia global es, por lo tanto:

$$Y=(X+Z)/2$$
$$X=Y-T$$
$$T=(Z-X)/2$$
$$Z=Y+T$$

Mediante esta agenda se puede siempre determinar la solución global.

Por ejemplo, la solución global a este problema ICSP

$$\{X:=1, T:=5, Y:=6, Z:=11\}$$

se obtiene a partir de la solución local

$$\{X:=1, T:=[0,10], Y:=[1,11], Z:=11\}.$$

Debe notarse que la longitud de la agenda en la propagación de la tolerancia global es menor que en el caso local debido a que para los parámetros del ciclo las funciones locales correspondientes al ciclo se han reemplazado por una única función global. Sin embargo, la funciones solución globales tienen expresiones funcionales más complicadas.
En el ejemplo que acabamos de presentar, las funciones solución globales se pueden derivar y utilizar de forma fácil. Sin embargo, en el caso general aparecen dos problemas importantes:

(a) *Limitaciones Algebraicas.* Las funciones solución globales no se pueden normalmente resolver por medios algebraicos. Este es un problema común a todos los sistemas de razonamiento algebraico. Sin embargo, en nuestro enfoque la dificultad de manipulación de una red algebraica se reduce porque es suficiente con obtener las funciones solución globales sólo con respecto a los conjuntos de restricciones cíclicas correspondientes a los ciclos. Dichos conjuntos son, en general, aunque no siempre, menores que el conjunto de la red de restricciones. Además, es posible deducir y utilizar sólo algunas funciones globales durante la propagación de la tolerancia dependiendo de las limitaciones del solver de problemas algebraicos disponible. Lo cual daría una solución mejor que la solución local pero peor que las global. Debe observarse también que incluso en el caso de algunos ICSP cíclicos la propagación local de la tolerancia llega a determinar la solución global. Por ejemplo, utilizando la propagación de la tolerancia local se puede determinar la solución del problema ICSP altamente cíclico siguiente

$$Z = X * Y$$
$$e^X + Z = -0.2$$
$$\sin(-Z) + X + Y = -0.4$$

(b) *Evaluaciones aritméticas algebraicas.* Además de resolver los problemas de ecuaciones algebraicas, la evaluación numérica de funciones intervalares comporta algunos problemas específicos de los intervalos.

A continuación, los problemas (1) y (2) son discutidos con más detalle y se presentan técnicas adicionales basados en la aritmética intervalar para resolverlos.

- **Análisis Intervalar**

Un problema clave en la propagación de la tolerancia global es como evaluar el rango de valores de funciones solución intervalares globales anidadas $F(X_1, \cdots, X_n)$. La principal dificultad que se presentan cuando se lleva a cabo dicha evaluación es que la aritmética intervalar difiere de la aritmética de valores exactos en algunas leyes algebraicas básicas. Por ejemplo, la ley distributiva $I*(J+K) = I*J+I*K$ no se cumple en la aritmética intervalar. Dicha aritmética sólo cumple la **ley subdistributiva**:

$$I(J+K) \subseteq IJ + IK$$

Así por ejemplo:

$$[1,2]*([-1,1]+[1,2]) = [0,6]$$
$$\subseteq [1,2]*[-1,1]+[1,2]*[1,2] = [-1,6]$$

Como consecuencia de ello, diferentes extensiones intervalares F de una misma función de valores exactos f puede evaluar diferentes valores de intervalos con los mismos argumentos incluso si F y f fueran equivalentes desde el punto de vista de valores exactos. Si la extensión F evalúa el rango real de valores se denomina **extensión óptima**.

Funciones intervalares básicas como las operaciones básicas (+,-,*,/) son óptimas. En cambio, los problemas aparecen cuando dichas funciones se combinan en formas anidadas como las que se ha presentado anteriormente.

Se han desarrollado diferentes formas para representar y evaluar funciones intervalares F. La más simple es la **extensión natural** que se obtiene reemplazando las variables reales por las variables intervalares en la función restricción real f. La extensión natural no es necesariamente óptima. Sin embargo, se puede demostrar que es monótona inclusiva. Esto significa que si $F(X_1,\cdots,X_n)$ es una extensión natural finita basada en variables intervalares $X_1,\cdots,X_n$ y las operaciones de la aritmética intervalar óptimas básicas, entonces

$$X_1' \subseteq X_1, X_2' \subseteq X_2, \cdots, X_n' \subseteq X_n$$
$$\text{implica} \quad F(X_1',\cdots,X_n') \subseteq F(X_1,\cdots,X_n)$$

para todo conjunto de intervalos $X_1,...,X_n$ para el cual el conjunto de operaciones de la aritmética intervalar estén definidos en F. Intuitivamente, la extensión natural garantiza la evaluación de un superconjunto ($\subseteq$) de valores reales. Si se restringen los intervalos de los argumentos, la extensión proporciona también valores más restringidos. Sin embargo, a menos que no se cumpla la igualdad $F(X_1',\cdots,X_n') = F(X_1,\cdots,X_n)$, F no es óptima. Las extensiones naturales de las funciones solución globales obtenidas utilizando las reglas algebraicas ordinarias se pueden por lo tanto aplicar durante la propagación de tolerancia global sin perder la completitud o consistencia del razonamiento. De esta forma, se pueden obtener intervalos más estrechos que utilizando propagación de tolerancia local pura, aunque la consistencia global no se puede garantizar a menos que se utilicen las funciones óptimas.

Existe una clase importante de funciones intervalares que se sabe que son óptimas:

***Teorema 10.3**: "Si la expresión de una función intervalar* $Y = F(X_1,\cdots,X_n)$ *no presenta múltiples instancias de las variables (o si las variables con múltiples instancias tienen valores exactos, lo que las convierte en constantes), entonces F es óptima si las operaciones básicas están definidas y son óptimas".*

Así por ejemplo, las funciones solución

$$Y = (X+Z)/2$$
$$X = Y-T$$
$$T = (Z-X)/2$$
$$Z = Y+T$$

del ejemplo que estabamos utilizando anteriormente no presentan múltiples instancias de las variables, por lo tanto, son óptimas y determinan la intervalo solución global. El Teorema 10.3 que acabamos de presentar admite la siguiente interpretación topológica: las funciones intervalares que no presentan múltiples instancias de las variables se corresponden a redes de restricciones acíclicas.

Desafortunadamente, no es normalmente posible obtener las funciones solución globales (óptimas) sin múltiples instancias de las variables. En estos casos, la red de restricciones no converge necesariamente hacia la solución global, incluso peor, normalmente no es posible obtener las funciones globales debido a las limitaciones del álgebra, a menos que se trate con ecuaciones algebraicas simples como las lineales. Sin embargo, es siempre posible determinar (por eliminación de las otras variables cíclicas) al menor una función recursiva global de la forma

$$X = F(\cdots, X, \cdots)$$

donde el valor de la variable X aparece probablemente una o más veces en la expresión funcional de F. A fin de determinar el rango global real de los valores de X se precisa un análisis intervalar más profundo.

En el campo de la aritmética intervalar, se han desarrollado técnicas numéricas y algebraicas para evaluar los valores óptimos o al menos más precisos de las funciones intervalares que puedan ser obtenidos mediante la evaluación directa de la extensión natural. Dichas técnicas también pueden ser aplicadas a funciones recursivas del tipo presentado anteriormente. A continuación presentamos una revisión de las mismas:

A. Técnicas Numéricas

Existe una forma, teóricamente simple y siempre aplicable, de determinar numéricamente el rango real de valores de funciones intervalares. Supongamos una función intervalar $F(X_1,\cdots,X_k,X_{k+1},\cdots,X_n)$, donde $X_1,\cdots,X_k$ aparecen más de una vez y $X_{k+1},\cdots,X_n$ aparecen sólo una vez en F. Dividamos cada $X_i = [a, b]$ para i = 1, ..., k en m subintervalos D_i:

$$D_i = \{[a, x_1],[x_1, x_2],\cdots,[x_{m-1}, a]\} \qquad (10.36)$$

Entonces $F(X_1,\cdots,X_k,X_{k+1},\cdots,X_n)$ se puede calcular de la siguiente manera:

$$F(X_1,\cdots,X_k,X_{k+1},\cdots,X_n) = \cup\{F(Y_1,\cdots,Y_k,X_{k+1},\cdots,X_n) | Y_i \in D_i,\quad i=1,\cdots,k\} \qquad (10.37)$$

Skelboe [Skelboe74] muestra que considerando $X_1,\cdots,X_k$ en partes cada vez más pequeñas, es decir, incrementando el valor de m, se pueden obtener aproximaciones arbitrariamente próximas del valor real de F. Esta técnica resuelve, teóricamente, el problema del cálculo de los valores de la función solución global. Sin embargo, la aplicación directa de dicho método significa que F debería ser evaluada m^k veces, lo cual es computacional muy costoso. Un algoritmo simple para reducir el número de evaluaciones de la función se presenta en [Skelboe74], refinada en [Moore79] y más desarrollada en [Asaithambi82]. La técnica se puede extender a funciones recursivas de la forma presentada anteriormente mediante la iteración de los valores de X convergentes hasta que se alcanza un punto fijo con una determinada precisión.

El problema que presentan también los métodos numéricos anteriores es que son especialmente ineficientes si los valores de F convergen lentamente, es decir, si F está lejos del óptimo. A continuación, presentamos formas de $F(X_1,\cdots,X_n)$ más óptimas y con una mayor velocidad de convergencia que la que se puede obtener con la extensión natural.

B. Técnicas Algebraicas

Un problema central al que se enfrenta la investigación en el campo de la aritmética intervalar es el siguiente: Cual es la extensión más óptima y con mayor velocidad de convergencia para representar una función intervalar?. En general, esta cuestión aún no está resuelta pero se han desarrollado diversas formas útiles de representar extensiones intervalares. Algunas de las formas más conocidas se presentan a continuación:

(1) *Modificaciones Algebraicas.* La extensión intervalar natural de una función se puede modificar utilizando las propiedades de la aritmética intervalar. Por ejemplo, utilizando la propiedad subdistributiva es siempre más útil utilizar la forma I*(J+K) en lugar de I*J+I*K. Así, por ejemplo, para polinomios $y = \sum a_i x^i$, por ejemplo, $y = ax^3 + bx^2 + cx + d$, la correspondiente extensión intervalar utilizando dicha propiedad $y = d + x(c + x(b + ax))$, denominada Esquema de Horner, normalmente produce unos intervalos más estrechos y además con menor esfuerzo computacional que la extensión natural.

(2) *Extensiones del Valor Medio y de Taylor.* La extensión intervalar del valor medio de una función F_m está basada en el teorema del valor medio. Para una función de una única variable F(X), está extensión es la siguiente $F_m(X = f(c) + F'(X)(X - c)$ donde *c* es el centro de *X*, o en general, $c \in X$. La extensión del valor medio de una función de n variables $F(X_1, \ldots, X_n)$ se define de la siguiente

manera $F_m(X) = f(c) + F_i'(X)(X_i - c)$ para i = 1,...,n y donde X es el vector $(X_1,...,X_n)$ y $F_i'(X)$es el vector de derivadas parciales de F con respecto a X_i. Se ha demostrado que la extensión del valor medio es también inclusiva monótona y se puede obtener fácilmente para el caso de funciones racionales e irracionales. A menudo, pero no siempre, dicha extensión obtiene unos intervalos más estrechos que la extensión natural. La extensión del valor medio se utiliza como la base del método de Newton intervalar que se puede utilizar para evaluar el rango de una función con la precisión deseada. La extensión del valor medio $F_m(X)$ es el caso de orden k =1 de una extensión más general, la extensión de Taylor $T_k(X)$, basa el desarrollo en serie de Taylor de una función

$$T_k(X) = f(c) + \sum \frac{f^{(\lambda)}(c)}{\lambda!}(X-c)^{\lambda} + \frac{F^{(k)}(X)}{k!}(X-c)^{k} \qquad (10.38)$$

donde $f^{(n)}(x)$ indica la n-ésima derivada de la función f(x). Para el caso de una función multidimensional existe un desarrollo análogo.

(3) *Extensiones centradas*. La extensión centrada de una función $F(X_1,...,X_n)$ se define mediante

$$F_c(X_1,\cdots,X_n) = f(c_1,\cdots,c_n) + G(X_1 - c_1,\cdots,X_n - c_n) \qquad (10.39)$$

donde c_i es el centro de X. La idea de la extensión centrada es desplazar el origen en c_i de forma que el valor de los argumentos de la función caigan alrededor de 0, lo cual minimiza los valores absolutos de los límites intervalares, y las funciones intervalares deben evaluar intervalos de menor tamaño. Las extensiones centradas normalmente proporcionan mejores resultados que las extensiones naturales, pero no son necesariamente óptimas ni inclusivas monótonas.

Las diferentes extensiones intervalares puede evaluar rangos radicalmente diferentes para una función en un determinado intervalo. Por ejemplo, para la función polinómica

$$y = x^4 - 10x^3 + 35x^2 - 50x + 24$$

al utilizar las diferentes extensiones intervalares presentadas hasta el momento para evaluar el rango de dicha función en el intervalo X:=[0,4] se han obtenido los siguientes valores

Extensión	Valor
Natural	[-816,840]
Horner	[-256,384]
Valor medio	[-116,116]
Centrada	[-24,36]
Exacta	[-1,24]

Tal como se observa en la tabla anterior ninguna de las extensiones obtiene el rango exacto, o sea, el óptimo. Las extensiones están ordenadas en dicha tabla en orden de mayor precisión con respecto a dicho ejemplo. Experimentalmente, este orden parece que se mantiene al menos para funciones de tipo polinómico, pero en general el orden puede ser diferente para diferentes tipos de función.

Hasta el momento, no existe un método universal para determinar la extensión intervalar óptima, sino que debe evaluarse la función utilizando diferentes extensiones y utilizar la intersección de las soluciones para determinar la mejor aproximación. La mejor manera de determinar el rango de valores de una función es combinar técnicas algebraicas con técnicas numéricas, es decir, determinar una extensión intervalar eficiente y aplicar entonces el método de Skelboe o Hansen sobre la misma para evaluar con la precisión deseada el rango de la función. La evaluación de funciones intervalares complicadas mediante técnicas numéricas es a menudo computacionalmente costoso. Por lo tanto, es aconsejable cuando se utiliza el algoritmo de propagación de tolerancia global colocar dichas funciones al final de la agenda.

C. Condiciones de Aplicación

Las técnicas de aritmética intervalar presentadas en los apartados anteriores suponen que las funciones intervalares están acotadas y definidas dentro de los intervalos donde se evalúan. Por ello, en general se deberán formular las condiciones de aplicación para la aplicación del algoritmo de propagación de tolerancia global. En el caso del algoritmo de propagación de tolerancia local, dichas condiciones se podían determinar fácilmente para funciones algebraicas básicas debido a su simplicidad algebraica. La determinación de dichas condiciones para funciones con expresiones anidadas es un poco más complicado. Por ejemplo, dada la función

$$X = A / (B - C * D)$$

puede ser aplicado sólo si $0 \notin (B - C * D)$.

El problema de la determinación de las condiciones de aplicación del algoritmo de propagación de tolerancia global se puede resolver mediante la evaluación de cada función básica $F(X_1,...,X_n)$ en una expresión anidada mediante

$$F(D_1 \cap X_1, \cdots, D_n \cap X_n) \qquad (10.40)$$

donde D_i para i=1,...,n es la división del espacio local de aplicación del argumento X_i y las funciones se evalúan utilizando la aritmética de divisiones presentada en secciones anteriores. Por ejemplo, supongamos que la función anterior está compuesta de tres funciones básicas anidadas:

$$\{X = A / X_1, X_1 = B - X_2, X_2 = C * D\}$$

La evaluación de las condiciones de aplicación para la situación

$$\{A:=1,\ B:=[1,2],\ C:=[-1,3],\ D:=[-2,0^-|0^+,2]\}$$

se realizaría de la siguiente manera:

Función	Condición
$X_2 = C * D = [-6,6]$	Ninguna condición: $D_c = D_D = [-\infty, \infty]$
$X_1 = B - X_2 = [-5,8]$	Ninguna condición: $D_B = D_{x_2} = [-\infty, \infty]$
$X = A/(X_1 \cap [-\infty, 0^- \mid 0^+, \infty] = [-\infty, -1/5 \mid 1/8, \infty]$	$D_A = [-\infty, \infty]$, $D_{x_1} = [-\infty, 0^- \mid 0^+, \infty]$

Después de la evaluación de una función básica anidada, el operador unión de la aritmética de divisiones no debería de ser aplicado: se obtienen resultados más óptimos aplicándolo sólo una vez después de evaluar la función más externa.

Así mismo, la siguiente técnica de localización de funciones solución puede ser utilizada en la evaluación de funciones intervalares. La función solución global anidada Y = F(...) puede ser representada como una red de restricciones S. Si S es acíclica, es decir, Y = F(...) no tiene múltiples incidencias de una misma variable, entonces si se añaden las funciones solución de S en la agenda y se aplica el algoritmo de propagación de la tolerancia local, P contendrá el valor del intervalo real de F(...). Sin embargo, es suficiente con añadir a la agenda sólo aquellas funciones solución locales de Y = F(...) que correspondan a sus funciones anidadas explícitas. Si Y = F(...) tiene múltiples instancias de la misma variable, entonces el problema de determinar el valor real de Y se puede representar recursivamente como un problema ICSP con la red Y = F(...). De esta forma el problema de resolver un problema ICSP globalmente se pude reducir a pequeños problemas ICSP si Y = F(...) tiene menos restricciones que el problema ICSP original.

Resumiendo, las actuales técnicas numéricas de la aritmética intervalar hacen posible evaluar el rango real de una función intervalar con la precisión deseada. Esto significa que evaluando las funciones solución global de un problema ICSP utilizando dichas técnicas, se pueden determinar las soluciones de

un problema ICSP con la precisión deseada. La eficiencia de dichas técnicas dependen de la extensión intervalar utilizada para representar el problema ICSP y sus funciones solución globales, así como el algoritmo de evaluación numérico utilizado.

10.8 Solvers de Problemas con Restricciones Intervalares: ICE InC++

A continuación presentamos la librería ICE InC++, implementación práctica de los algoritmos de la propagación de la tolerancia presentados en este capítulo para resolver problemas de satisfacción de restricciones intervalares. Dicha librería se diferencia de los softwares basados en "constraint logic programming" en:

- ICE InC++ no descompone las restricciones en restricciones funcionales primitivas. De esta forma, las restricciones individuales se pueden resolver globalmente. Los softwares que utilizan descomposición sólo pueden determinar soluciones locales, obteniendo por lo tanto unos intervalos resultado con una anchura mayor.
- Desde el punto de vista práctico, un problema importante de los softwares basados en "constraint logic programming" es su dificultad de integración con las posibles aplicaciones que se puedan realizar, puesto que dichos softwares normalmente están basados en lenguajes tipo PROLOG de difícil integración con aplicaciones realizadas en lenguajes más estándares como C++. En cambio, ICE InC++ se ha realizado en C++ en forma de librería fácilmente portable entre diferentes sistemas evitando de esta manera los posibles problemas de integración y eficiencia.

Mediante la librería ICE InC++, un problema de satisfacción de restricciones intervalares se define mediante:

- un conjunto de ecuaciones y desigualdades
- el valor de los intervalos iniciales de las variables
- y opcionalmente, los parámetros de cálculo: tales como el nivel de precisión absoluta o relativa, el tiempo límite para el cálculo, etc.

La librería tiene dos modos de funcionamiento:

- En el modo de acotación de la solución ("solution bounding mode"), la salida es un conjunto de valores de intervalos obtenidos mediante el refinamiento del valor de los intervalos iniciales, o bien, la indicación de la infactibilidad del problema de satisfacción de restricciones intervalares. Se pueden ejecutar diversas operaciones de relajación para aumentar los intervalos de cara a convertir un problema infactible en uno factible.
- En el modo de determinación de la solución, se obtienen 1...n soluciones exactas, o bien, se detecta la infactibilidad del problema.

Si los cálculos se interrumpen debido al tiempo límite de cálculo establecido, la librería devuelve los resultados parciales obtenidos hasta ese momento.

La librería ICE InC++ utiliza una combinación de técnicas basadas en:

- análisis intervalar,
- optimización global,
- álgebra computacional y
- propagación de la tolerancia.

ICE se caracteriza por ser un sistema de filtrado local, que evalúa les restricciones individuales independientemente unas de otras. Sin embargo, ICE InC++ determina cotas más estrechas que los sistemas de filtrado local tradicionales debido a que posee las siguientes características avanzadas:

- *Aritmética Intervalar Extendida.* ICE InC++ utiliza una aritmética intervalar extendida en la cual los intervalos pueden ser multintervalos discontinuos. Una situación intervalar con multintervalos se corresponde a un conjunto de situaciones continuas. Durante el proceso de resolución de restricciones esta subsituaciones se pueden considerar separadamente mediante un esquema de búsqueda, lo cual en general proporciona cotas más estrechas que el filtrado simple.
- *Evaluación Global de Restricciones.* Los valores de las variables durante el proceso de evaluación de las restricciones se determinan mediante rutinas de optimización intervalar globales, siguiendo los algoritmos propuestos por Ratschek y Rokne [Ratschek84][Ratschek88] y por Hansen [Hansen92]. Para cada restricción, se puede determinar el verdadero rango de valores para cada variable que posea una sola instancia. Para variables que posean múltiples instancias dentro de la restricción el rango que se determina es aún sobredimensionado. ICE In C++ no descompone las expresiones de las restricciones en restricciones primitivas. De esta forma las dependencias entre instancias de variables pueden ser capturadas obteniéndose resultados más ajustados.
- *Optimización Algebraica.* ICE In C++ hace uso también de optimización algebraica, para intentar transformar las restricciones en una forma algebraicamente más adecuada para que se le aplique el análisis intervalar.

El corazón de la librería ICE InC++ es la clase *Ice* que se utiliza para representar un problema de satisfacción de restricciones intervalares. Se utiliza un conjunto de métodos asociados con esta clase para insertar/eliminar restricciones, establecer los valores de las variables, establecer los parámetros de cálculo, evaluar las funciones y leer los resultados. Un objeto *Ice* se mantiene automáticamente internamente coherentes después de las modificaciones introducidas por el usuario.

La aplicación de la clase *Ice* en un programa en C++ conlleva realizar los siguientes pasos:

Utilización de la librería ICE InC++

1. Crear una instancia Ice, a la que denominaremos por ejemplo E.
2. Insertar/Eliminar las ecuaciones o desigualdades intervalares asociadas al problema de satisfacción de restricciones intervalares.
3. Establecer los valores para las variables que forman parte del problema de satisfacción de restricciones intervalares.
4. Modificar, opcionalmente, los parámetros de cálculo, por ejemplo, los niveles de precisión deseados del objeto Ice si es necesario.
5. Evaluar E y leer los resultados.
6. Repetir los pasos 3,4 y 5 si es necesario.

10.9 Aplicación a la Generación de Envolventes

El problema de generación de envolventes utilizando el algoritmo propuesto en esta tesis a la vista de la teoría expuesta en este capítulo se puede formular como un problema de satisfacción de restricciones intervalares. En primer lugar, vamos a recordar de nuevo la formulación de dicho algoritmo.

10.9.1 Formulación del problema de satisfacción de restricciones

Básicamente el algoritmo consistía en determinar para cada una de las variables de estado del sistema el valor máximo y mínimo en el instante de tiempo actual, teniendo en cuenta el valor de las variables de estado L instantes hacia atrás, donde L era longitud de la ventana temporal, y teniendo en cuenta que los parámetros del sistema eran constante entre instantes de tiempo pero de valor incierto pero acotado en un intervalo. La formulación matemática de dicho algoritmo era la siguiente:

$x_n^+ = \max\left(A^L x_{n-L} + A^{L-1}Bu_{n-L} + \cdots + Bu_{n-1}\right)$ *s.t.:* $x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$	$x_n^- = \min\left(A^L x_{n-L} + A^{L-1}Bu_{n-L} + \cdots + Bu_{n-1}\right)$ *s.t.:* $x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$

que de una forma más compacta se puede expresar como:

$x_n^+ = \max x_n = \max\left(A^L x_{n-L} + \sum_{i=n-L}^{n-1} A^{n-1}Bu_i\right)$ *s.t.:* $x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$	$x_n^- = \min x_n = \min\left(A^L x_{n-L} + \sum_{i=n-L}^{n-1} A^{n-1}Bu_i\right)$ *s.t.:* $x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$ $A \in \left[A^-, A^+\right]$ $B \in \left[B^-, B^+\right]$

que a su vez se puede reescribir como un solo problema de satisfacción de restricciones intervalares de la siguiente manera:

Restricciones: $x_n = A^L x_{n-L} + \sum_{i=n-L}^{n-1} A^{n-1}Bu_i$

Variables: $x_n \in (-\infty,+\infty)$, $x_{n-L} \in \left[x_{n-L}^-, x_{n-L}^+\right]$, $A \in \left[A^-, A^+\right]$, $B \in \left[B^-, B^+\right]$

Es decir, el problema tendrá tantas restricciones como variables de estado tenga el sistema, cada restricción se obtendrá a partir de la expresión matemática que proporciona el valor de un estado en el instante actual en función del valor de los estados al inicio de la ventana y en función del valor de los parámetros del sistema. Así, para un sistema con *k* variables de estado, $\vec{x} = (x_1, \cdots, x_k)$ tendremos k restricciones una para cara variable de estado:

$$E_1: \quad x_{1n} = f(\vec{x}_{n-L}, A, B)$$
$$\cdots$$
$$E_k: \quad x_{kn} = f(\vec{x}_{n-L}, A, B)$$

en cada una de estas k restricciones aparecerán una de las variables de estado en el instante actual, las *k* variables de estado en el instante L instantes hacia atrás, o sea, al inicio de la ventana y los parámetros del sistema. Por lo tanto, este problema de satisfacción de restricciones intervalares tendrá como variables las siguientes:

- variables asociadas a las variables de estado al instante actual

$$P_1 := x_{1n}, \ \ldots, \ P_k := x_{kn}$$

- variables asociadas a las variables de estado al instante al inicio de la ventana

$$P_{k+1} := x_{1(n-L)}, \ldots, P_{2k} := x_{k(n-L)}$$

- variables asociadas a los parámetros de la matriz A del sistema

$$P_{2k+1} := a_{11}, \ldots, P_{2k+k*k} := a_{kk}$$

- variables asociadas a los parámetros de la matriz B del sistema

$$P_{2k+k*k+1} := b_1, \ldots, P_{2k+k*k+k} := b_k$$

Para cada una de estas variables el intervalo de valores asociado es el siguiente:

- variables asociadas a las variables de estado al instante actual

$$X_1 := [x_{1n}^-, x_{1n}^+], \ldots, X_k := [x_{kn}^-, x_{kn}^+]$$

- variables asociadas a las variables de estado al instante al inicio de la ventana

$$X_{k+1} := [x_{1(n-L)}^-, x_{1(n-L)}^+], \ldots, X_{2k} := [x_{k(n-L)}^-, x_{k(n-L)}^+]$$

- variables asociadas a los parámetros de la matriz A del sistema

$$X_{2k+1} := [a_{11}^-, a_{11}^+], \ldots, X_{2k+k*k} = [a_{kk}^-, a_{kk}^+]$$

- variables asociadas a los parámetros de la matriz B del sistema

$$X_{2k+k*k+1} = [b_1^-, b_1^+], \ldots, X_{2k+k*k+k} = [b_k^-, b_k^+]$$

Una vez formulado el problema de generación de envolventes como un problema de satisfacción de restricciones intervalares, deberemos resolverlo con alguno de los métodos para resolución de este tipo de problemas. Puesto que los intervalos que estamos buscando para cada unas de las variables de estado en el instante actual son los intervalos exactos, deberemos utilizar un método que nos pueda garantizar que los intervalos encontrados son **globalmente consistentes**. El único método que puede garantizar tal consistencia es el método de la propagación de tolerancia global propuesto por Hyvönen [Hyvönen] y que se ha expuesto anteriormente.

Dicho algoritmo está disponible mediante la librería ICE InC++ desarrollo tal como hemos visto por el grupo del propio Hyvönen. La utilización de dicha librería también se ha presentado en la sección anterior.
La aplicación del algoritmo de propagación de la tolerancia global para resolver el problema de satisfacción de restricciones intervalares que se ha presentado en este capítulo a la resolución del problema de optimización planteado al utilizar el nuevo algoritmo de generación de envolventes que se propone en esta tesis pasa por:

Paso 1. Inicialización. Determinar la expresión de las funciones restricciones para cada uno de los estados del sistema del cual se quieren obtener las envolventes. Esta tarea se puede realizar con cualquier programa de manipulación simbólica como MAPLE.
Paso 2. Formulación del problema de satisfacción de restricciones intervalares de forma que el solver basado en la propagación de la tolerancia global, ICE InC++, pueda resolverlo.
Paso 3. Solución del problema de satisfacción de restricciones intervalares utilizando el solver ICE InC++.

Paso 4. Lectura de los resultados del problema de satisfacción de restricciones intervalares y generación de las envolventes en el instante de tiempo actual.
Paso 5. Generación del nuevo problema de satisfacción de restricciones intervalares, para obtener el valor de las envolventes en el siguiente instante de tiempo, actualizando las cotas de los estados al inicio de la ventana temporal, a partir de los resultados obtenidos en la solución del problema asociado al instante actual.
Paso 6. Salto al paso 3.

Gráficamente, esta secuencia de pasos se puede representar de la siguiente manera:

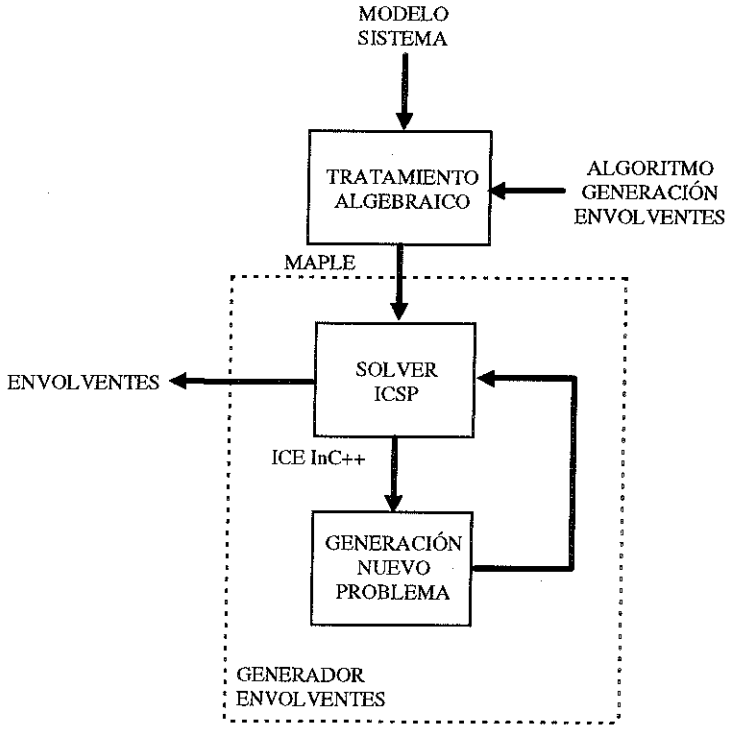


Fig. 10.8 Generación de las envolventes utilizando ICE InC++

10.9.2 Aplicación a un sistema de segundo orden

El sistema a partir del cual generaremos las envolventes utilizando el nuevo algoritmo de generación presentado en esta tesis y para calcular las envolventes a cada instante resolveremos el algoritmo utilizando el método de propagación de la tolerancia global para problemas de satisfacción de restricciones intervalares propuesto por Hyvönen.

El modelo discreto del sistema de segundo orden en el espacio de estado es el siguiente

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

con los siguientes parámetros del modelo inciertos

$$a_1 \in [1.3938, 1.4338]$$

$$a_2 \in [0.5865, 0.6265]$$

Si aplicamos el nuevo algoritmo de generación de envolventes a este sistema y formulamos el problema resultante como un problema de satisfacción de restricciones intervalares obtendremos, para el caso que escojamos una ventana temporal L = 5,

Restricciones:

$$E_1: \quad x_{1k} = x_{1(k-L)}a_1^5 - 4x_{1(k-L)}a_1^3a_2 + 3x_{1(k-L)}a_1a_2^2 - a_2x_{2(k-L)}a_1^4 + 3a_2^2x_{2(k-L)}a_1^2 - a_2^3x_{2(k-L)} + a_1^4 - 3a_1^2a_2 + a_2^2 + a_1^3 - 2a_1a_2 + a_1^2 - a_2 + a_1 + 1$$

$$E_2: \quad x_{2k} = x_{1(k-L)}a_1^4 - 3x_{1(k-L)}a_1^2a_2 + x_{1(k-L)}a_2^2 - a_1^3a_2x_{2(k-L)} + 2a_1a_2^2x_{2(k-L)} + a_1^3 - 2a_1a_2 + a_1^2 - a_2 + a_1 + 1$$

Variables:

P_1: $x_{1k}=(-\infty,+\infty)$

P_2: $x_{2k}=(-\infty,+\infty)$,

P_3: $x_{1(k-L)}=[\underline{x}_{1(k-L)}, \overline{x}_{1(k-L)}]$

P_4: $x_{2(k-L)}=[\underline{x}_{2(k-L)}, \overline{x}_{2(k-L)}]$

P_5: $a_1=[1.3938,1.4338]$

P_6: $a_2=[0.5865,0,6265]$

In Fig. 10.9 y Fig. 10.10, se presenta la representación gráfica de las funciones restricción correspondientes a las restricciones E_1 y E_2.

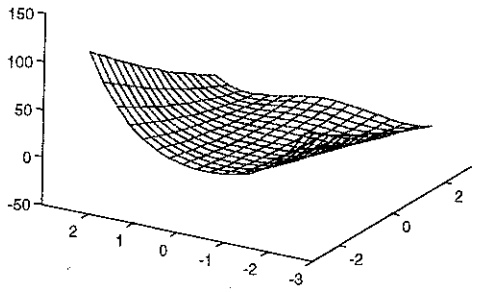


Fig. 10.9 Representación gráfica de la función restricción E_1 para una ventana de longitud L=5

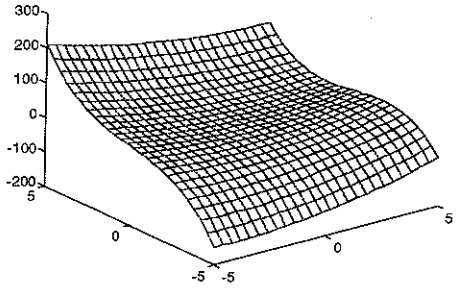


Fig. 10.10 Representación gráfica de la función restricción E_2 para una ventana de longitud L=5

Para resolver el problema de satisfacción de restricciones intervalares utilizaremos la librería ICE InC++ que implementa el algoritmo de propagación de la tolerancia global.

Los resultados obtenidos con este solver al resolver el problema de satisfacción de restricciones intervalares asociado al problema de generación de envolventes sobre el ejemplo del sistema de segundo orden presentado en esta sección para diferentes longitudes de ventana y empezando siempre en el instante inicial k=0 son los que se muestran en la Tabla 1. Las envolventes calculadas corresponden a las variables de estado x_1 y x_2.

L	Envolvente para el estado x_1	Envolvente para el estado x_2	Tiempo Cálculo
5	[5.18649,6.12189]	[4.67125,5.16882]	0.17 seg
10	[4.27974,7.05731]	[4.53032,7.16635]	9 seg
15	[4.22014,6.51874]	[4.14801,6.57872]	900 seg
20	[4.33664,6.51730]	[4.34751,6.50320]	28000 seg

Tabla 1. Envolventes para las variables de estado x_1 y x_2

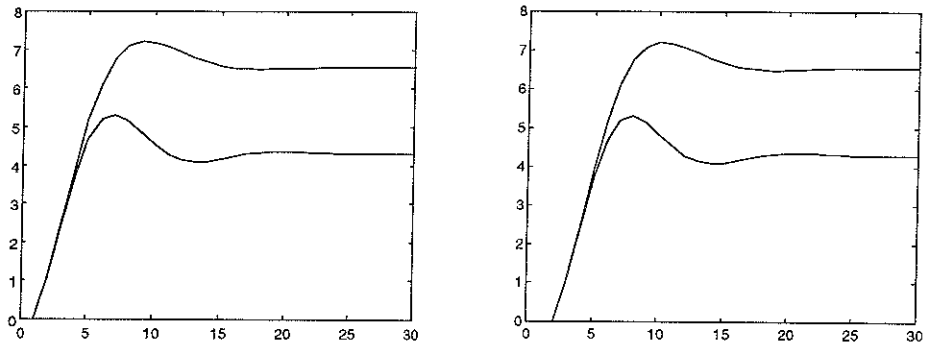
A partir de estos resultados, se observa que para una ventana mayor que L=15 el tiempo de cálculo aumenta de forma que resulta de poco interés para aplicaciones reales. Por ejemplo, para L =20 el tiempo de cálculo que hace falta para obtener las envolventes es 28000 seg, que equivalen aproximadamente a 8 horas de cálculo. Todos estos resultados se han obtenido utilizando ICE InC++ con una precisión de 10^{-6}, suficiente para considerar los resultados como exactos. Evidentemente, si se hubiera exigido una precisión menor el tiempo de cálculo sería menor.

A partir de los resultados obtenidos en el capítulo 3 de esta tesis, podemos determinar cual es la longitud mínima estable y cual es la longitud mínima estacionaria. Recordemos, que la longitud mínima estable es la longitud de la ventana necesaria para evitar un crecimiento incontrolado de la incertidumbre debido al algoritmo de generación de envolventes utilizado, mientras que la segunda longitud de ventana se corresponde con la longitud necesaria para producir aproximadamente, con sólo un incremento en la incertidumbre total del 5%, los mismo resultados que se obtendrían si los cálculos de las envolventes en el instante de tiempo actual se refirieran siempre al instante inicial k = 0, que tal como hemos visto este es la única forma de generar las envolventes sin aumentar la incertidumbre. Aplicando las expresiones obtenidas en el capítulo 3, podremos obtener analíticamente las anteriores longitudes de ventana, para el ejemplo del sistema de segundo orden que estamos utilizando

Mínima Estable	$L_\infty=7$
Casi Estacionaria	$L_{0.05}=15$

Tabla 2. Longitudes de ventana teóricas

A partir de estos cálculos, observamos que la ventana de longitud 15 produce las envolventes del sistema con un incremento del 5% en la incertidumbre presente en el sistema, respecto al caso exacto. En la Fig. 10.11 se muestran las envolventes para las variables de estado x_1 y x_2 utilizando la longitud de ventana L = 15 utilizando el método de satisfacción de restricciones intervalares presentado en este capítulo.



(a) Variable de Estado x_1 (b) Variable de Estado x_2

Fig. 10.12 Envolventes para las variables de estado x_1 y x_2 a partir del nuevo algoritmo de generación de envolventes y utilizando el método de satisfacción de restricciones intervalares con una longitud de ventana(L=15)

10.10 Conclusiones

En este capítulo se ha presentado una nueva técnica para resolver el problema de optimización asociado con el problema de generación de envolventes mediante el nuevo algoritmo de generación de envolventes presentado en esta tesis. Esta nueva técnica a diferencia de las técnicas presentadas en los capítulos anteriores no trata de resolver el problema como si fuera un problema de optimización sino que trata de resolverlo como un problema de satisfacción de restricciones, y en concreto de restricciones intervalares. Existen un conjunto de técnicas para resolver este tipo de problemas, de entre las cuales en este capítulo nos hemos centrado en la propuesta por Hyvönen y que se denomina de propagación de la tolerancia. De la misma forma que los problemas de optimización tienen el problema de la existencia de soluciones óptimas locales que podían hacer que si se utiliza un algoritmo de búsqueda local basado en un punto semilla a partir del cual se inicia la búsqueda utilizando la información que proporcionan las derivadas alcanzara uno de estos puntos óptimos locales y quedara atrapado en él y finalizara dando como solución dicho punto, con lo cual no se puede tener la certeza si la solución alcanzada es la óptima global o es una de local, los problemas de satisfacción de restricciones tienen también un problema parecido el denominado problema de la consistencia local. Es decir, muchos algoirtmos de resolución de problemas de satisfacción de restricciones se detienen cuando alcanza la consistencia local, no pudiendo garantizar en general, salvo en determinados casos concretos, donde las restricciones cumplen unas determinas propiedades de aciclidad, que se alcazará la consitencia global. Sin embargo, el algoritmo de propagación de tolerancia de Hyvönen combinado con los algoritmos de optimización global basados en la búsqueda branch and bound y el álgebra de intervalos puede garantizar la tolerancia global.

El hecho de que se proponga resolver el problema de optimización asociado al problema de generación de envolventes como un problema de satisfacción de restricciones intervalares se debe tal como se ha comentado a lo largo de este capítulo a que se trata de un enfoque mucho más general, puesto que el problema de generación de envolventes se traduce en un único problema de satisfacción de restricciones

donde cada una de las ecuaciones que proporcionan las envolventes para uno de los estados del sistema se puede ver como una de las restricciones del problema.

Sin embargo, si comparamos los resultados obtenidos utilizando como método de resolución del problema de generación de envolventes las técnicas de satisfacción de restricciones frente a las técnicas de optimización global basadas en la búsqueda de tipo branch and bound presentadas en el capítulo anterior, observamos que si bien los resultados a nivel de precisión que se pueden obtener con ambas técnicas son las mismos, el tiempo de cálculo necesario para obtener los mismos resultados con la misma precisión es mucho más elevado cuando se utilizan las técnicas de satisfacción de restricciones implementadas en ICE InC++ que si utilizamos las técnicas de optimización global implementadas en GIA InC++. Ellos nos conduce a que el método a utilizar para aplicaciones en tiempo real en las que se deseen utilizar las envolventes como método de detección deberemos de utilizar las técnicas de optimización global de GIA InC++.

Si bien la utilización de las técnicas de satisfacción de restricciones permiten una formulación más compacta del problema de generación de envolventes que la que proporcionan las técnicas de optimización global, puesto que resolviendo un sólo problema de satisfacción de restricciones obtenemos directamente todas las envolventes del sistema, mientras que si utilizamos las técnicas de optimización global deberemos de resolver un problema de optimización global para cada una de las variables de estado del sistema. Ello podría nos podría llevar a pensar que en este caso el tiempo total debería de ser mayor que para el caso de la resolución del único problema de satisfacción de restricciones intervalares, pero en la práctica se demuestra que el tiempo necesario para este segundo caso es mayor debido a que el tipo de problema de satisfacción de restricciones intervalares que debemos de resolver estará compuesto por dos restricciones que a su vez deberán de ser resueltas con técnicas de optimización global al contener múltiples instancias de la misma variable. Ello conlleva que el tiempo como mínimo sea igual al de resolver los dos problemas de optimización global directamente utilizando el solver de optimización global, pero además a ese tiempo se le debe añadir el tiempo de preparación del problema y la aplicación del algoritmo de propagación de la tolerancia global.

Capítulo 11

Capítulo 11:

INTEGRACIÓN DEL ALGORITMO DE GENERACIÓN DE ENVOLVENTES CON ALGORITMOS DE DIAGNÓSTICO DE FALLOS

En este capítulo se pretende dar una perspectiva de como el algoritmo de generación de envolventes pueden integrarse fácilmente dentro de un sistema de detección y diagnóstico de fallos basado en modelos. En definitiva, este es el capítulo de aplicación de la generación de envolventes a la finalidad para la cual al inicio de esta tesis había sido pensada: la generación de umbrales adaptativos en el contexto de la detección y diagnóstico de fallos en procesos industriales. En este capítulo vamos a presentar todos los ingredientes necesarios para llegar a construir un sistema de detección y diagnóstico de fallos robusto: vamos a presentar una variante de un algoritmo de diagnóstico de fallos ya existente el algoritmo DMP completándolo con los algoritmo de detección de fallos basados en la generación de residuos y evaluación de los mismos mediante umbrales adaptativos generados mediante envolventes. El guión de este capítulo podría resumirse en los pasos a seguir para implementar un sistema de detección y diagnóstico de fallos siguiendo este nuevo algoritmo DMP que lo hemos denominado mejorado:

- Modelado del Sistema con Fallos
- Generación de los Residuos a partir de las Ecuaciones de Paridad
- Generación de los Umbrales Adaptativos (Envolventes)
- Toma de Decisión mediante Lógica Fuzzy
- Generación de Residuos Estructurados
- Matriz de Sensibilidad de los Residuos frente a los Fallos
- Vector de Fallos

11.0 Modelado de la Dinámica del Sistema con Fallos

Aunque los procesos industriales son normalmente procesos continuos, los algoritmos de diagnóstico de fallos trabajan normalmente con datos discretos o muestrados. Por lo tanto, normalmente se utiliza un modelo discreto para describir la dinámica del sistema. Así mismo, se supone que los modelos utilizados serán modelos lineales, de forma que en el caso de trabajar con un sistema cuya dinámica sea no lineal, el modelo lineal se obtendrá a partir de la linealización de su dinámica alrededor del punto de trabajo.

- **Ecuaciones del Sistema**

Consideremos un sistema con m entradas y p salidas

$$\begin{aligned} u(k) &= [u_1(k), \cdots, u_m(k)] \\ y(k) &= [y_1(k), \cdots, y_p(k)] \end{aligned} \qquad (11.1)$$

donde k es la variable que representa el tiempo discreto. Entonces el modelo del sistema expresado en el espacio de estado puede expresarse como

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \qquad (11.2)$$

donde x(k) es el vector de estado con dimensión n y A,B,C y D son matrices constantes. Análogamente, la dinámica del mismo sistema se puede expresar mediante un modelo del tipo entrada-salida descrito mediante

$$y(k) = S(z)u(k) \qquad (11.3)$$

donde cada elemento de la matriz S(z) es una función de transferencia, o sea, una función racional del operador desplazamiento z. El mismo sistema puede también escribirse como

$$H(z)y(k) = G(z)u(k) \qquad (11.4)$$

donde H(z) y G(z) son matrices polinómicas en z^{-1}

$$\begin{aligned} G(z) &= G_o + G_1 z^{-1} + \cdots + G_n z^{-n} \\ H(z) &= I + H_1 z^{-1} + \cdots + H_n z^{-n} \end{aligned} \qquad (11.5)$$

siendo H(z) una matriz diagonal. El modelo entrada-salida está relacionada con el modelo en el espacio de estado mediante

$$S(z) = C(zI - A)^{-1}B + D \qquad (11.6)$$

por lo tanto

$$\begin{aligned} G(z) &= C\mathrm{Adj}(I - z^{-1}A)z^{-1}B + D \\ H(z) &= [\mathrm{Det}(I - z^{-1}A)]I \end{aligned} \qquad (11.7)$$

En la práctica, el modelo entrada-salida casi nunca se obtiene a partir del modelo en el espacio de estado, sino que se obtiene directamente de la física del sistema y mediante identificación de parámetros.

- **Modelado de los Fallos**

A partir del modelo en el espacio de estado, los fallos se modelan normalmente de la siguiente manera

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + Fp(k) \\ y(k) &= Cx(k) + Du(k) + q(k) \end{aligned} \qquad (11.8)$$

donde u(k) y y(k) son las observaciones, o sea, el valor de control para las entradas y el valor medido por los sensores para las salidas, p(k) y q(k) son los vectores de fallo y F es la matriz de fallos. El vector p(k) puede ser de cualquier dimensión: sus elementos representan fallos en actuadores, determinados fallos en la planta, perturbaciones y fallos en los sensores que miden las entradas. Por otro lado el vector q(k) contiene los fallos en los sensores que miden las salidas. La matriz de fallos F se supone conocida. Los fallos son modelados como funciones desconocidas en el tiempo y normalmente no se hace ninguna suposición sobre su comportamiento temporal.

Al transformar la representación del sistema con fallos en el espacio de espacio al modelo entrada-salida, introduciremos una representación más detallada del modelo de los fallos que nos permitirá profundizar en la naturaleza de los diferentes fallos. Incluyendo los fallos en la planta y las perturbaciones junto con las entradas y descomponiendo el vector de entradas observables u y el vector de fallos asociados Δu según

$$u = \begin{bmatrix} u_M \\ u_C \\ u_D \end{bmatrix} \qquad \Delta u = \begin{bmatrix} \Delta u_M \\ -\Delta u_C \\ -\Delta u_D \end{bmatrix} \qquad (11.9)$$

donde:

u_M son los valores medidos para las entradas medidas y Δu_M son los fallos en los sensores de entrada.
u_C son los valores de control para las entradas controladas y Δu_C son los fallos en los actuadores.
u_D es el valor de las perturbaciones de entrada y fallos en la planta y Δu_D es su valor actual

Estos valores están relacionadas con los verdaderos valores, es decir, con los valores reales de la planta de acuerdo con

$$\begin{aligned} u_M^o &= u_M - \Delta u_M \\ u_C^o &= u_C + \Delta u_C \\ u_D^o &= \Delta u_D \\ y^o &= y - \Delta y \end{aligned} \qquad (11.10)$$

tal como se muestra en la Fig. 11.1.

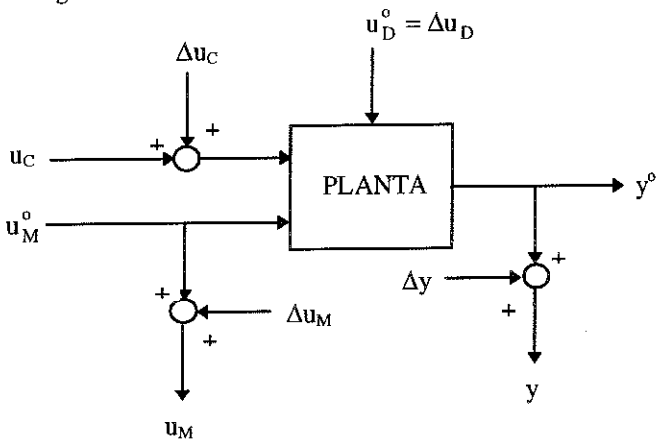


Fig. 11.1 Modelo de la planta con fallos

Las ecuaciones que representan el modelo entrada-salida relacionan las variables reales de la planta. Utilizando las relaciones que relacionan las variables reales con los fallos obtendremos las siguientes relaciones

$$\begin{aligned} y(k) &= S(z)u(k) + \Delta y(k) - S(z)\Delta u(k) \\ H(z)y(k) &= G(z)u(k) + H(z)\Delta y(k) - G(z)\Delta u(k) \end{aligned} \qquad (11.11)$$

En ambas ecuaciones, los dos primeros términos representan las observaciones mientras que los otros dos contienen los fallos y las perturbaciones. Tal como hemos comentado anteriormente, no se ha realizado ninguna suposición sobre el comportamiento temporal de los fallos y las perturbaciones.

Debe observarse que sólo se han considerado fallos y perturbaciones aditivas. Aunque en la mayoría de los casos es suficiente, en determinados casos puede ser necesario considerar también fallos multiplicativos. Dichos fallos aparecen debido a cambios en parámetros de la planta. Lo que hace difícil su tratamiento es que los coeficientes en las ecuaciones del modelo no son constantes, sino que que contienen variables de la planta.

11.1 Generación de Residuos mediante Ecuaciones de Paridad

- **Ecuaciones de paridad a partir del modelo entrada-salida**

A partir del modelo de la planta con fallos

$$y(k) = S(z)u(k) + \Delta y(k) - S(z)\Delta u(k) \qquad (11.12)$$

en su forma explícita, o bien,

$$H(z)y(k) = G(z)u(k) + H(z)\Delta y(k) - G(z)\Delta u(k) \qquad (11.13)$$

en su forma implícita.

Los residuos simplemente se definen como

$$\begin{aligned} e(k) &= H(z)y(k) - G(z)u(k) \\ &= \Delta y(k) - S(z)u(k) \end{aligned} \qquad (11.14)$$

a partir del **modelo explícito.**

O bien

$$\begin{aligned} e^*(k) &= H(z)y(k) - G(z)u(k) \\ &= H(z)\Delta y(k) - G(z)u(k) \end{aligned} \qquad (11.15)$$

a partir del **modelo implícito** .

Ambas ecuaciones son **ecuaciones de paridad** que poseen las siguientes propiedades:

1. En ambas ecuaciones, la primera línea corresponde a la **forma computacional** (es decir, contiene a las variables observables) mientras que la segunda línea corresponde a la **forma interna** (es decir, contiene a los fallos).
2. **e(k)** depende de las variables observables y los fallos a través de una función de transferencia. Estas ecuaciones de paridad son del **tipo ARMA**, o tipo **Output Error**. Mientras que **e*(k)** dependen de las variables observadas y los fallos a través de polinomios. Estas ecuaciones son del **tipo MA**, o tipo **Polynomial Equation Error**. Debe observarse que la conversión entre los formatos ARMA y MA sólo requiere la multiplicación/división a través del común denominador de las funciones de transferencia.
3. En ambas ecuaciones, cada residuo escalar depende sólo de una de las variables de salida (en la forma computacional) y en una de los fallos de la salida (en la forma interna). Dichas ecuaciones/residuos se denominarán **ecuaciones de paridad o residuos primarios**.

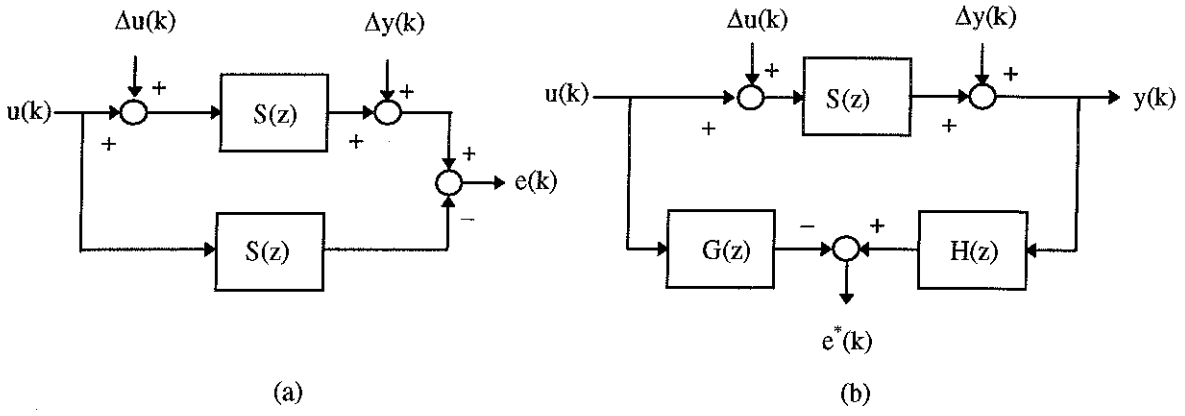


Fig. 11.2 Ecuaciones de Paridad: (a) Tipo ARMA (explícitas) y (b) Tipo MA(implícitas)

Las ecuaciones de paridad multiplicadas o divididas por polinomios o funciones racionales de z y combinadas son también ecuaciones de paridad. Por lo tanto, ecuaciones residuales adicionales pueden ser generadas a partir del conjunto primario mediante transformación lineal

$$\begin{aligned} r(k) &= V(z)e(k) \\ r^*(k) &= W(z)e^*(k) \end{aligned} \qquad (11.16)$$

donde los elementos de la matriz de transformación V(z) son funciones racionales de z y las ecuaciones de paridad resultantes se encuentran también en el formato ARMA mientras que los elementos de W(z) son polinomios en z y las ecuaciones resultantes se encuentran en formato MA. El diseño de un conjunto de ecuaciones de paridad consiste en seleccionar la matriz de transformación de tal manera que los residuos r(k) o $r^*(k)$ posean ciertas propiedades deseadas.

Debe remarcarse que cada ecuación de paridad de tipo ARMA es explícita para una de las salidas y que las ecuaciones ARMA transformadas se pueden siempre bajo esta forma. Dicha forma corresponde a la interpretación más natural de una relación de consistencia, en la cual el valor actual de una salida se compara con el valor estimado por el modelo a partir del resto de variables, según se muestra en la Fig. 11.3.

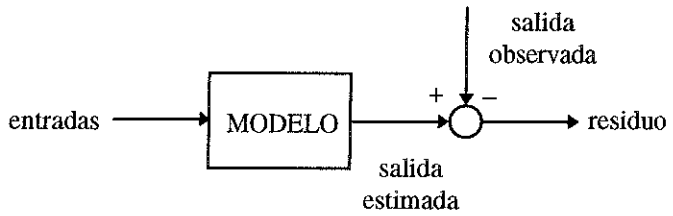


Fig. 11.3 Generación del residuo mediante la salida estimada y observada

11.2 Robustez en los Residuos: Umbrales Adaptativos

Todos los modelos de plantas están sujetos a errores de modelado. Dichos errores aparecen debido a:

- las imprecisiones iniciales del modelado
- derivas lentas debidos al paso del tiempo, fugas, etc.
- variaciones cíclicas debidas al ciclo de trabajo de la planta, cambios en la temperatura, etc.

Los errores de modelado pueden afectar las estructura del modelo (aproximación de orden inferior de sistemas de orden superior) y a sus parámetros.

El generador de residuos se construye a partir del modelo nominal de la planta, de forma que cualquier discrepancia entre el mismo y la planta real conducirá a la obtención de residuos diferentes de cero en ausencia de fallos. Por lo tanto, los errores de modelado pueden interferir gravemente con los algoritmos de detección y diagnosis que se estén utilizando. Por lo tanto, conseguir que los generadores de residuos sean insensibles a los errores de modelado es de gran importancia. Este es el problema al que las técnicas de detección robustas se enfrentan.

Consideremos las ecuaciones del sistema en el espacio de estado

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \qquad (11.17)$$

bien, análogamente en la forma entrada-salida

$$H(z)y(k) = G(z)u(k) \qquad (11.18)$$

si introducimos, en las ecuaciones anteriores $(A^\circ, B^\circ, C^\circ, D^\circ)$ y $(H^\circ(z), G^\circ(z))$, respectivamente, para describir el comportamiento nominal de la planta y $(\Delta A^\circ, \Delta B^\circ, \Delta C^\circ, \Delta D^\circ)$ y $(\Delta H^\circ(z), \Delta G^\circ(z))$ para describir los errores de modelado, por lo tanto, la relación con el sistema verdadero (A, B, C, D) y $(H(z), G(z))$, valdrá respectivamente

$$\begin{aligned} A^{\circ} &= A + \Delta A \\ B^{\circ} &= B + \Delta B \\ C^{\circ} &= C + \Delta C \\ D^{\circ} &= D + \Delta D \end{aligned} \qquad (11.19)$$

para el caso del modelo en el espacio de estado, mientras que para el modelo entrada-salida

$$\begin{aligned} G^{\circ}(z) &= G(z) + \Delta G(z) \\ H^{\circ}(z) &= H(z) + \Delta H(z) \end{aligned} \qquad (11.20)$$

Si ahora utilizamos el modelo nominal del sistema en la forma entrada-salida para generar los residuos, los residuos producidos por los errores de modelado valdrán:

$$e^{*}(k) = \Delta H(z)y(k) - \Delta G(z)u(k) \qquad (11.21)$$

Analógamente utilizando el modelo en el espacio de estado para generar el residuo, concretamente, si utilizamos un observador

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + K[y(k) - Cx(k)] = (A - KC)x(k) + Bu(k) + Ky(k) \\ r(k) &= y(k) - Cx(k) \end{aligned} \qquad (11.22)$$

Si definimos el error de estimación como $e(k) = \hat{x}(k) - x(k)$ entonces la expresión del observador anterior se puede reformular como

$$\begin{aligned} e(k+1) &= (A - KC)e(k) \\ r(k) &= Cx(k) \end{aligned} \qquad (11.23)$$

Por lo tanto, los residuos debidos a los errores de modelado valdrán, para este caso

$$\begin{aligned} e(k+1) &= (A^{\circ} - K^{\circ}C^{\circ})e(k) - (\Delta A - K^{\circ}\Delta C)x(k) - \Delta Bu(k) \\ r(k) &= C^{\circ}e(k) - \Delta Cx(k) \end{aligned} \qquad (11.24)$$

En ambos casos, se observa que a diferencia de las ecuaciones que describían los residuos que aparecerían en presencia de fallos aditivos, ahora las matrices son desconocidas mientras que las variables que las multiplican son conocidas o observables. Se trata de una diferencia sustancial que lleva a que el problema de robustez frente a los errores de modelado sea intratable con los tratamientos utilizados para desacoplar el efecto de las perturbaciones de tipo aditivo presentados hasta ahora.

Desafortunadamente, no existe una solución completa al problema de generación de residuos robustos. Por lo tanto, no es posible construir un generador de residuos que sea completamente inmune a todos los errores de modelado a la vez que mantiene su sensibilidad frente a los fallos. Las soluciones parciales que en la literatura se sugieren para afrontar el problema de la robustez en al generación de los residuos siguen alguna de las siguientes estrategias:

- Suponer que la incertidumbre en el modelo se puede caracterizar por un conjunto finito de variantes de modelos. Entonces es posible diseñar un generador de residuos individual que funciones razonablemente bien para cualquiera de las variantes contenida en el conjunto. Esta estrategia fue propuesta por Lou [Lou86] y Frank and Wunnenberg [Frank89].
- Suponer que todos los errores de modelado se pueden deducir a partir de la incertidumbre de un conjunto de parámetros. Entonces, se pueden calcular las derivadas parciales de los residuos respecto a dichos parámetros, de forma que se pueden construir generadores de residuos en línea con

sensibilidad parcial igual a cero, o bien, generadores de residuos con la menor sensibilidad posible escogidos a partir de un conjunto precalculado. Esta estrategia fue propuesta por Gertler [Gertler90].

Por otro lado, la utilización de residuos estructurados para el caso de fallos aditivos proporciona un cierto grado de robustez inherente frente a los errores de modelado. Ello se debe a que los errores de modelado difícilmente producirán exactamente el mismo código de error que el que producirá uno de los fallos posibles.

Un enfoque alternativo a la detección robusta de fallos implica calcular, en línea o fuera de línea, los residuos que pueden aparecer bajo ciertas suposiciones de errores de modelado y fijar los umbrales de test de acuerdo con los residuos así calculados, dichos umbrales se denominan umbrales adaptativos. Estas técnicas tal como hemos comentado en los primeros capítulos se denominan técnicas pasivas. Este enfoque fue propuesto por Horak [Horak88] para el caso de considerar la incertidumbre estructurada y Emami-Naeni [Emami88] para el caso de considerar la incertidumbre como desestructurada.

La generación de umbrales adaptativos utilizando un modelo con incertidumbre estructurada se realiza mediante la generación de las envolventes utilizando alguno de los algoritmos de generación de las mismas. En nuestro caso, utilizaremos el nuevo algoritmo de generación de envolventes presentado en esta tesis.

- **Generación de umbrales adaptativos utilizando envolventes**

La detección de fallos en sistemas dinámicos se basa en la comparación de la respuesta del sistema monitorizado con la estimación de dicha respuesta utilizando un modelo del sistema. La comparación se puede realizar directamente sobre las medidas, o bien indirectamente utilizando una cantidad calculada tal como las innovaciones de un estimador. Si se dispone de un modelo exacto del sistema monitorizado, incluso la menor diferencia entre las respuestas medida y estimada constituye una indicación fiable de la existencia de un fallo. Puesto que en esta situación ideal, el modelo del sistema no contiene incertidumbre sobre la dinámica del sistema real, el test de la existencia de un fallo puede utilizar un umbral cero.

En cambio, si existen errores de modelado, el modelo no representa la dinámica del sistema exactamente. Consideremos la siguiente descripción general de un sistema dinámico con incertidumbre en el modelo

$$\begin{aligned} \dot{x}(t) &= [A_n + A_e]x(t) + [B_n + B_e]u(t) \\ y(t) &= [C_n + C_e]x(t) + [D_n + D_e]u(t) \end{aligned} \qquad (11.25)$$

donde las matrices A_n, B_n, C_n y D_n representan el modelo nominal del sistema, mientras que las matrices intervalares A_e, B_e, C_e y D_e representan la incertidumbre sobre los parámetros del modelo. Cada uno de los elementos de dichas matrices es un intervalo definido por una cota inferior y superior. Así el valor del elemento a_{ij} de la matriz del sistema A, se encuentra dentro del intervalo $a_{nij} - a_{eij} \le a_{ij} \le a_{nij} + a_{eij}$, donde $2a_{eij}$ es la anchura del intervalo.

En algunos sistemas, la incertidumbre intervalar en los parámetros de las ecuaciones del modelo se pueden relacionar con la tolerancia de los componentes y su valor se puede determinar analíticamente. Si no se dispone del modelo analítico del sistema, la incertidumbre intervalar en los parámetros se debe determinar de forma experimental. Una forma fácil es realizar una estimación de los mismos utilizando las técnicas de identificación de sistemas. Entonces los intervalos de incertidumbre asociados a los parámetros se pueden obtener a partir de las varianzas del valor de los mismos proporcionadas por el propio método de estimación.

El modelo del sistema con incertidumbre en los parámetros tal como se ha descrito mediante la ecuación anterior sólo dispone de los rangos en los que se encuentra cada uno de los parámetros del sistema cuando no está en fallo, pero no se dispone de su valor exacto. Por lo tanto, el sistema de detección de fallos basado en dicho modelo deberá comparar la respuesta obtenida a partir de las medidas del sistema

real con un gran número de respuestas posibles obtenidas variando los parámetros inciertos del modelo del sistema dentro de su rango de valores posibles. De forma que el sistema de detección de fallos deberá de indicar la existencia de un fallo si la respuesta medida no se puede justificar a partir de ninguna de las respuesta estimadas a partir del modelo, y concretamente, por ninguna de las correspondientes a la peor combinación de parámetros, o sea, a las envolventes del sistema.

El sistema de detección de fallos no debe producir falsas alarmas frecuentes. Por lo tanto, puesto que los valores reales de los parámetros del sistema son desconocidos, el test de detección debe suponer que nos encontramos frente a la peor combinación de parámetros posible, de forma que el umbral de detección será lo suficientemente grande para evitar falsas alarmas. Sin embargo, dicha consideración introduce conservadurismo en el test de detección, debido a que la desviación real de los parámetros no siempre será la peor posible la mayoría de las veces. Como consecuencia de ello, el uso de un umbral que es demasiado grande para los valores reales de los parámetros del sistema se traducirá en fallos no detectados. Este conservadurismo es inevitable si se desean evitar las falsas alarmas, siendo el precio a pagar por el hecho de desconocer el valor exacto de los parámetros.

El mejor test de detección en sistemas con parámetros inciertos es aquel que no introduce más conservadurismo del propio debido a la incertidumbre en los parámetros. Dicho test utiliza para cada instante de tiempo el menor umbral posible para evitar falsas alarmas en presencia de la peor desviación posible en los parámetros del sistema. La especificación del mejor test es única, por lo tanto, existe un único test que cumpla dicha especificación y sus prestaciones delimitan el límite de la detectabilidad de los fallos. Dicho test se corresponde con la definición de las envolventes del sistema. Por lo tanto, la generación de las envolventes de un sistema nos proporcionará el mínimo umbral posible para cada instante de tiempo.

El mínimo umbral posible es igual a la máxima variación posible de la desviación de una medida (o de una función de las medidas) de su valor calculado a partir del modelo nominal del sistema. Este valor óptimo del umbral garantiza que incluso para la peor desviación posible en los parámetro que no es considerada como fallo, o sea, aquella que produce la mayor desviación en las medidas, no producirá una falsa alarma. El mínimo umbral posible se puede calcular de forma exacta mediante la simulación a partir del modelo del sistema, empezando en el instante inicial y finalizando en el instante actual, a la vez que se varían los valores de los parámetros inciertos para conseguir el máximo valor posible en el valor de las medidas. El algoritmo de generación de envolventes presentado en esta tesis, determina los valores de los parámetros inciertos que maximizan las desviaciones en las medidas sin tener que estudiar todas las combinaciones posibles de los parámetros. El algoritmo determina así mismo los parámetros que minimizan los valores de las medidas. El intervalo definido por el valor máximo y mínimo para cada instante de tiempo, la envolvente, es el intervalo de medidas alcanzable. Dicho intervalo incluye todos las medidas que el sistema sin fallo podría llegar a generar. El intervalo define la menor región umbral posible: su anchura es la suma del menor umbral posible para desviaciones positivas y negativas de las medidas respecto a su valor nominal. La región umbral menor posible representa el conservadurismo inevitable de cualquier test de detección de fallos que utilice un modelo con errores de modelado en los parámetros.

Debido a que el menor test posible depende de las entradas y estados pasadas y actuales, dicho test debe recalcular el menor umbral posible a cada instante de tiempo. Cualquier método de detección de fallos que no recalcule el menor umbral posible a cada instante de tiempo añade mayor conservadurismo al proceso de detección y funcionará peor que el test óptimo que hemos definido, siendo incapaz de detectar fallos tan pequeños como los que detecta el test óptimo sin producir frecuentes falsas alarmas.

11.3 Toma de Decisión bajo Incertidumbre

Una vez generado el vector de residuos, utilizando cualquiera de las técnicas de generación de residuos presentadas hasta el momento, se debe analizar la información contenida en el mismo. Esta fase de denomina fase de toma de decisión. El vector de residuos contiene información se los fallos, pero desafortunadamente también sobre el ruido, la perturbaciones existentes en el sistema y los errores de modelado. En ausencia de ruido, perturbaciones y errores de modelado, el vector de residuos mostraría después de un análisis lógico o un método de clasificación, cual es el fallo o fallos existentes en el sistema real. En la práctica existirá ruido, perturbaciones y errores de modelado, por lo tanto, en la toma

de decisión se deberá de tener en cuenta su efecto y será necesario separarlo del efecto que producen los fallos sobre los residuos.

Para analizar, el vector de residuos bajo la presencia de incertidumbre debida al ruido, las perturbaciones y los errores de modelado se dispone de las siguientes técnicas:

- **Métodos estadísticos**, que trabajan con cada elemento del vector de residuos como si fuera una variable aleatoria de media cero, en ausencia de fallos. La declaración de un residuo como significativo se realiza en función de los parámetros estadísticos que definen un comportamiento, en la práctica los más utilizados son la media y la desviación típica.
- **Métodos deterministas**, que trabajan con cada elemento del vector de residuos determinando cuál es el valor máximo y mínimo que puede tomar teniendo en cuenta la información disponible sobre los errores de modelado y el ruido. Estas técnicas se conocen también como técnicas de generación de umbrales adaptativos.

Una vez determinado el valor umbral, ya sea por métodos estadístico, o bien, deterministas se debe decidir si se ha violado dicho umbral y si dicha violación es consecuencia de la existencia de un fallo. Para tomar dicha decisión existen las siguientes técnicas:

- **Método de los dos umbrales**, consistente en establecer dos umbrales, uno superior y otro inferior, sobre el parámetro estadístico que se elija, normalmente la media o la desviación típica, y un contador del número de veces que cada residuo sobrepasa dichos umbrales. Si sobrepasa el umbral superior el contador se incrementará en 1 y si se el que se sobrepasa es el inferior se decrementará en una unidad. Adicionalmente se establecerá un límite en el contador que será el que determine si ese residuo es declarado significativo o no. El paso siguiente sería crear para cada residuo un vector de decisión binario que valdría 0 si el contador está por debajo del límite y 1 en los momentos que esté por encima. De esta forma, un fallo será detectado si este conjunto de vectores binarios coincide con el patrón de este fallo.

- **Método basado en la lógica fuzzy**. En la etapa de decisión, de una forma u otra, se compara el valor del residuo con un umbral. Cuando el umbral (ya sea fijo o adaptativo) es un valor numérico la decisión es muy sensible a pequeñas perturbaciones. Una alternativa a utilizar un umbral numérico es utilizar un umbral difuso. La utilización de umbrales difusos en la etapa de decisión permite evitar los efectos de la incertidumbre y de las falsas alarmas reduciendo el número de falsas alarmas y de alarmas no indicadas.

  La "fuzzificación" de los residuos consiste en asociar su valor a un determinado conjunto difuso. En el contexto de diagnóstico de fallos, se puede interpretar como la "fuzzificación" del umbral de decisión. Vamos a explicarlo mediante un ejemplo. Supongamos que se ha generado un vector de residuos $r(u,y)$ con la ayuda de alguno de los métodos de generación de residuos visto hasta ahora.

  Para la detección y aislamiento de fallo el residuo debe cumplir las siguientes condiciones ideales evaluadas mediante *"**lógica binaria**"*:

$$\begin{cases} r = 0 & \text{para} \quad f = 0 \\ r \neq 0 & \text{para} \quad f \neq 0 \end{cases} \qquad (11.26)$$

  En la práctica, debido a la incertidumbre asociada al modelo y al ruido asociado en las medidas, es necesario asociar *"**umbrales binarios**"* mayores que cero para evitar falsas alarmas. Esto implica una reducción de la sensibilidad en la detección de fallos, y con la elección del umbral sólo se puede alcanzar un compromiso entre la sensibilidad del detector de fallos y la tasa de falsas alarmas.

  Para evitar este problema, se puede fuzzificar el residuo o evaluarlo mediante un *"**umbral difuso**"*, deforma que su evaluación se realice mediante *"**lógica fuzzy**"*. Llegados a este punto, en primer lugar se debe determinar la magnitud del residuo bajo condiciones normales de funcionamiento del proceso cuando sólo están presentes los efectos de los errores de modelado y el ruido en las medidas (las

denominadas "entradas desconocidas"). El conjunto difuso {cero} o {sin fallo} quedará caracterizado por una función de pertenencia definida de la siguiente manera:

$$\mu_{r1}(x): x \in X \rightarrow [0,1] \qquad (11.27)$$

donde $X=\{x_1,x_2,...,x_p\}$ es el universo de discurso y $\mu_{r1}(x)$ es la función de pertenencia asociada de la forma mostrada en la Fig. 11.4, donde el parámetro a_o debe asignarse de forma proporcional a la amplitud del ruido y los efectos de la incertidumbre en el modelo. El parámetro δ se debe escoger como la varianza del ruido presente en el proceso y de las influencias de los errores de modelado.

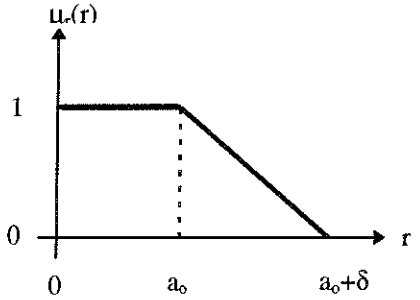


Fig. 11.4 Función de pertenencia del conjunto difuso {cero} bajo perturbaciones y incertidumbre en el modelo

De forma similar se puede definir el conjunto difuso {uno} o {con fallo}. Consideremos la Fig. 11.5. En la Fig. 11.5a se muestra el método convencional de evaluación del residuo con un umbral fijo. El máximo del residuo caracterizado por 1 puede representar una perturbación mientras que el máximo caracterizado por 2 puede ser debido a un fallo. Tal como se puede ver en dicha figura, el máximo de 1 no sobrepasa el umbral T, sin embargo, esto podría suceder con un pequeño incremento en la perturbación lo que conllevaría la aparición de una falsa alarma. De forma análoga, cambios infinitesimales en el segundo máximo podrían provocar la no detección del fallo.

Si ahora aplicamos sobre el umbral T un "suavizado" transformándolo en un intervalo de longitud finita de acuerdo con la Fig. 11.5b y definimos el conjunto {uno} mediante la función de pertenencia $\mu_{r2}(x)$ de acuerdo con la Fig. 11.5c. Esta modificación implica que un cambio pequeño en el valor del máximo del residuo 1 o del residuo 2 alrededor de T provocará un pequeño cambio que evitará la señalización de falsas alarmas o la no detección de fallos. El efecto de que pequeños causas se traducen en grandes efectos (inestabilidad estructural del sistema de diagnóstico de fallo) se pueden evitar mediante este "suavizado" del umbral de detección. Mediante la composición de los conjuntos difusos {cero} y {uno}, se obtiene el diagrama de pertenencia del residuo tal como se muestra en la Fig. 11.6. Evidentemente, la "fuzzificación" del umbral se puede interpretar directamente como la "fuzzificación" del residuo.

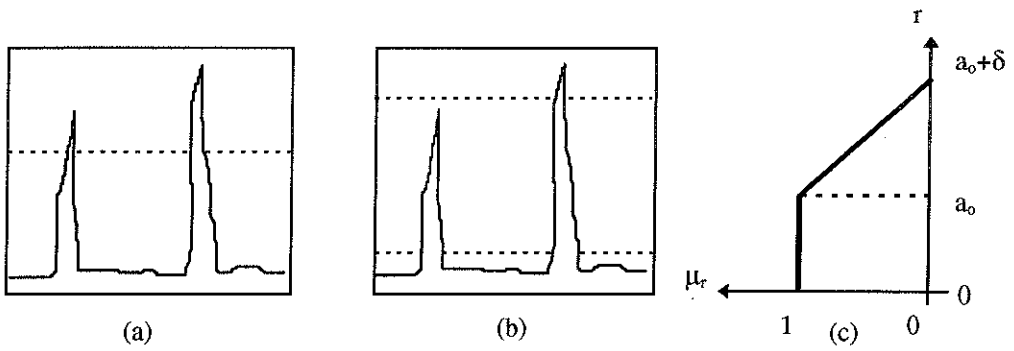


Fig. 11.5 Umbral de decisión binario vs umbral de decisión difuso

La función de pertenencia mostrada en la Fig.11.6 se puede considerar como la forma más simple de "fuzzificación" del umbral o del residuo mediante dos conjuntos difusos: {cero} y {uno}, o bien, {pequeño} y {grande}.

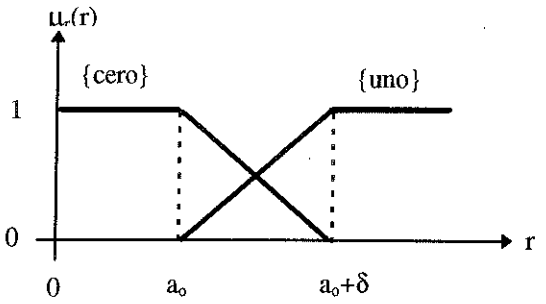


Fig. 11.6 "Fuzzificación" del residuo mediante dos conjuntos difusos {cero} y {uno}.

Esta "fuzzificación" puede extenderse a un número mayor de conjuntos difusos posibles: {pequeño}, {medio} y {grande}de acuerdo con la Fig. 11.7. La forma de la función de pertenencia se puede asignar mediante el conocimiento heurístico que se tenga del proceso, o bien, mediante función de distribución estadísticas, o bien, mediante aprendizaje utilizando redes neuronales (ajuste de los conjuntos y reglas difusas mediante redes neuronales).

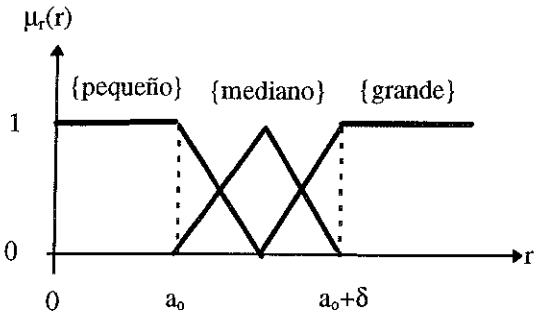


Fig. 11.7 "Fuzzificación" del residuo mediante tres conjuntos difusos {pequeño}, {mediano} y {grande}.

11.4 Residuos Estructurados

Mientras que un único residuo es suficiente para detectar un fallo, es necesario un conjunto de residuos para aislar un fallo. Para facilitar el aislamiento, las propiedades del conjunto de residuos deben ser mejoradas en alguna de las siguientes formas:

- En respuesta a un único fallo, sólo un subconjunto de residuos sensibles a este tipo de fallo deberán ser distintos de cero. Este enfoque se conoce como **residuos estructurados** y fue propuesto por Gertler and Singer [Gertler90] y Ben-Haim [Ben90].

- En respuesta a un único fallo, el vector residuo queda confinado en una dirección específica del fallo. Este enfoque se conoce como **residuos con direcciones prefijadas** y fue propuesto por Beard [Beard71].

Una de las formas de mejorar las propiedades de los residuos consiste en generar vectores de residuos r(k) de forma que, en repuesta a un determinado fallo p_j, o bien, q_j, sólo un determinado subconjunto de residuos sensible a mismo sea diferente de cero. En términos geométricos, $r(k|p_j)$ y $r(k|q_j)$ quedarán confinados a un subespacio del espacio de paridad determinado por un subconjunto de los vectores de

coordenadas. A dichos conjuntos de residuos se les denominará estructurados. Un conjunto estructurado cumple que cada residuo es completamente insensible a un subconjunto diferente de fallos.

La ventaja de los conjuntos de residuos estructurados es que el análisis de diagnosis consiste simplemente en determinar cuáles de los residuos son diferentes de cero. El test de umbral puede ser realizado sobre cada residuo de forma independiente, y consiste en una decisión booleana, donde un "1" indicada test positivo. Combinando el conjunto de estos bits de información en un vector binario se genera un **código o firma del fallo**. El conjunto de todos los posibles códigos de fallo nominales forma el **conjunto de códigos**.

Para la detección de todos los fallos, ningún código de fallo nominal debería contener todos los elementos iguales a cero. Un requerimiento mínimo para el aislamiento del fallo es que todos los códigos de fallo nominales sean distintos. Un código de fallo que satisfaga estos dos requisitos se le denomina **débilmente aislante**.

Bajo condiciones de test estadístico (entorno con ruido o errores de modelado), un código de fallo débilmente aislante no es suficiente. Los umbrales normalmente están fijados a valores altos y un fallo de tamaño moderado puede generar un código de fallo degradado, el cual algunos "1" han sido reemplazados por "0". Para evitar esta falta de aislamiento del fallo en dicha situación, el conjunto de códigos debería cumplir que ningún código degradado fuera idéntico a un código válido. Dicho tipo de conjunto de codificación se denomina **fuertemente aislante**.

Diversas estructuras especiales de residuos han sido propuestas en la literatura que pueden ser interpretadas dentro del formalismo de los residuos estructurados. Estas estructuras incluyen conjuntos donde cada residuo es afectado por:

- Todos los fallos excepto uno.
- Sólo uno de los fallos de los sensores
- Sólo uno de los fallos de los actuadores

El desacoplo de los residuos frente a las perturbaciones se puede manejar como un caso especial de residuos estructurados: sólo los residuos que no son afectados por la perturbación con incluidos en el conjunto.

Para satisfacer los requerimientos de aislamiento, es necesario o deseable generar vectores de residuos de $s > m$ dimensiones. Puesto que el número de salidas independientes es sólo m, existirán s-m relaciones entre los residuos en el espacio de paridad s-dimensional extendido.

- **Generación de los residuos estructurados a partir de las ecuaciones de paridad**

Los residuos estructurados pueden ser generados a partir de las ecuaciones de paridad estructuradas, tanto el formato ARMA como en el formato MA, según Gertler y Singer [Gertler85, Gertler90]. Una ecuación de paridad estructurada es aquella que está completamente desacoplada de un subconjunto fallos. Dichas ecuaciones se obtiene a partir del conjunto primario de ecuaciones residuales obtenidas mediante el procedimiento descrito anteriormente. Debe observarse que los residuos primarios e(k) y $e^*(k)$ son estructurados: cada residuo es afectado sólo por una de los fallos de salida, mientras que no es afectado por los m-1 fallos de salida restantes.
El procedimiento de transformación se puede aplicar de forma que cada vez se obtenga una ecuación de paridad. Para una ecuación de tipo MA

$$r_i^*(k) = w_i^T(z)[H(z)y(k) - G(z)u(k)] \qquad (11.28)$$

donde $w_i^T(z)$ es un vector de transformación de m polinomios cuyos elementos deben ser determinados. La ecuación objetivo se especifica en términos de su estructura. Para que $r_i^*(k)$ no resulte afectado por el fallo de una entrada Δu_j o por el fallo de una salida Δy_j, se requiere que

$$\begin{aligned} w_i^T(z)g_j(z) &= 0 \\ w_i^T(z)h_j(z) &= 0 \end{aligned} \qquad (11.29)$$

respectivamente, donde g_j y h_j son las matrices columna de G(z) y H(z) respectivamente. Cada una de las condiciones anteriores se corresponde con una ecuación homogénea para los m elementos de $w_i^T(z)$. Se debe satisfacer m-1 de tales condiciones para que $r_i^*(k)$ no se vea afectada por m-1 fallos, tanto de entrada como de salida.

En el proceso de transformación, todos los polinomios se manejan como entidades simples. Puesto que las ecuaciones son homogéneas, uno de los elementos de $w_i^T(z)$ debe ser elegido. Eligiéndolo adecuadamente, la solución será un polinomio. Se puede demostrar que el grado de las ecuaciones de paridad resultantes nunca excederá el orden del sistema n: si D = 0 entonces no excederá n - s_u donde s_u es el número de entradas eliminadas de la ecuación.
Las condiciones de existencia de las ecuaciones de paridad estructuradas están relacionadas con las propiedades de las columnas de la matriz G(z). Las propiedades más importantes son las siguientes:

Aislamiento mutuo. Si para dos columnas cualquiera $g_i(z)$ y $g_j(z)$ existen un par de polinomios $c_i(z)$ y $c_j(z)$ tales que

$$c_i(z)g_i(z)+c_j(z)g_j(z)=0 \qquad (11.30)$$

condición que implica dependencia lineal de polinomios, entonces no es posible crear ecuaciones de paridad que no estén afectadas por uno de los fallos Δu_i y Δu_j mientras que sea insensible al otro. En este caso, los dos fallos no son mútuamente aislables. El aislamiento mutuo es una propiedad fundamental del sistema: la no existencia de dicha propiedad sólo puede ser corregida mediante un cambio en el sistema físico, por ejemplo, añadiendo un sensor.

Alcanzabilidad. Para que una ecuación no se vea afectada por s_u entradas y s_y fallos en las salidas (s_u+s_y=m-1), se debe crear una matriz $(s_u+1)\times s_u$ $G_i(z)$, incluyendo las columnas de G(z) que pertenezca a las s_u entradas y excluyendo las filas que pertenezcan a las s_y salidas. Entonces las ecuaciones no son alcanzables si se cumplen alguna o ambas de las condiciones siguientes:

- alguna de las submatrices $s_u \times s_u$ de $G_i(z)$ pero no todas no son de rango completo
- cualquiera de las matrices $[G_i(z)|g_j(z)]$, donde $g_j(z)$ es una columna fuera de $G_j(z)$, no es de rango completo

Obsérvese que para el conjunto de ecuaciones de paridad residuales $r^*(k)$, las columnas de la matriz de incidencia [W(z)G(z)|W(z)H(z)] forman el conjunto de codificación.

11.5 Propiedades de la Matriz de Incidencia

Se define la **matriz de incidencia** Π para las ecuaciones de paridad obtenidas mediante

$$\begin{aligned} e^*(k) &= H(z)y(k)-G(z)u(k) \\ &= H(z)\Delta y(k)-G(z)u(k) \end{aligned} \qquad (11.31)$$

como una matriz booleana cuyos elementos están definidos de la siguiente manera:

$$\begin{aligned} \pi_{ij} &= 1 \quad \text{si} \quad f_{ij} \neq 0 \\ \pi_{ij} &= 0 \quad \text{si} \quad f_{ij} = 0 \end{aligned} \qquad (11.32)$$

donde $\pi_{ij}=1$ indica que una desviación en la medida de la variable j-ésima, originada por ruido o por un fallo, influye en el residuo i-ésimo, y $\pi_{ij}=0$ indica lo contrario. Si consideramos que cada fallo está

asociado directamente a una variable j-ésima, lo cual es evidente para los fallos en sensores y actuadores, el elemento π_{ij} indica si el fallo j-ésimo será detectado por el residuo i-ésimo. Un ejemplo de matriz de incidencia podría ser

	y_1	y_2	y_3	u_1	u_2	u_3	u_4
r_1	1	0	0	1	1	0	1
r_2	0	1	0	1	1	1	0
r_3	0	0	1	1	1	1	1

Para este caso, la declaración de r_1 y r_3 como residuos significativos y no de r_2, llevaría a la conclusión de que se ha producido un fallo asociado a la variable u_4, suponiendo que no se ha presentado un fallo múltiple. Sin embargo, los fallos asociados a las variables u_1 y u_2 no pueden diagnosticarse de forma tan inmediata para el sistema dado. La formalización de estas ideas intuitivas lleva a las siguientes definiciones:

- El conjunto de residuos obtenidos a partir de las ecuaciones de paridad tiene una **estructura sensible** al conjunto de fallos asociados a las variables observadas, entradas o salidas, si ninguna de las columnas de su matriz de incidencia Π tiene todos sus elementos nulos.
- Dicho conjunto de residuos tendrá una **estructura débilmente aislante** si todas las columnas de Π son distintas.
- Mientras que dicho conjunto de residuos tendrá una **estructura fuertemente aislante** si todas las columnas de Π tienen al menos dos elementos distintos entre sí.

Es decir, el conjunto de ecuaciones escogidas para describir el sistema tendrá una estructura sensible si podemos detectar todos los fallos como discrepancias entre el comportamiento teórico y el real. Además, si tiene estructura débilmente aislante, en ausencia de ruido se podrán diagnosticar todos sus fallos. Por último, una estructura fuertemente aislante, ofrecerá mayor robustez al ruido en el diagnóstico de fallos, e incluso permitirá detectar, en algunos casos, múltiples fallos. En general, la mínima distancia entre los patrones binarios que identifican cada fallo, columnas de la matriz Π, permite medir la capacidad de diagnóstico de un sistema, y por lo tanto su rosbustez frente a ruidos y perturbaciones.

Los modelos originales del sistema rara vez satisfacen las condiciones anteriormente citadas para su diagnosticabilidad. Por ello, puede ser de gran interés la transformación de la estructura del modelo en otra más adecuada. El nuevo modelo puede tener más, menos o las mismas ecuaciones que el original. Si fuesen más, algunas serían linealmente dependientes, pero la estructura de la matriz de incidencia sería distinta. La transformación del modelo puede verse como una recolocación de los ceros en las ecuaciones de paridad.

11.6 Matriz de Sensibilidad de los Residuos frente a los Fallos

La definición de sensibilidad de una función de transferencia G(s) con respecto al parámetro a fue introducida por Bode como:

$$S = \frac{d \ln G(s,a)}{d \ln a} = \frac{dG(s,a)}{da} \frac{a}{G(s,a)} \qquad (11.33)$$

Dicha definición pretende medir la influencia de pequeñas variaciones de un parámetro sobre una función de transferencia o sistema con realimentación. A continuación vamos a ver como se aplica al estudio de la sensibilidad de las ecuaciones de paridad residuales frente a la existencia de diferentes fallos.

- **Sensibilidad de las ecuaciones de paridad frente a los fallos**

Dada la ecuación matricial que describe el modelo entrada-salida del sistema en forma MA

$$H(z)y(k) = G(z)u(k) + Hz)q(k) + F(z)p(k) \qquad (11.34)$$

incluyendo los fallos en los en los sensores de salida q(k) y en las variables de entrada p(k). En particular, sea la ecuación de la salida i

$$\begin{aligned} h_i(z)y_i(k) = g_1^i(z)u_1(k) + g_2^i(z)u_2(k) + \cdots + g_n^i(z)u_n(k) + h_i(z)q_i(k) + \\ + f_1^i(z)p_1^i(k) + f_2^i(z)p_2^i(k) + \cdots + f_n^i(z)p_n^i(k) \end{aligned} \qquad (11.35)$$

El cálculo del residuo $r_i(k)$, se hará a partir de los parámetros estimados mediante identificación de parámetros:

$$r_i(k) = \hat{h}_i(z)\hat{y}_i(k) - \hat{g}_1^i(z)u_1(k) - \ldots - \hat{g}_n^i(z)\hat{u}_n(k) \qquad (11.36)$$

cuyo significado es

$$r_i(k) = h_i(z)q_i(k) + f_1^i(z)p_1^i(k) + \cdots + f_n^i(z)p_n^i(k) \qquad (11.37)$$

La medida de la sensibilidad de cada residuo $r_i(k)$ frente a cada fallo $f_j(k)$ se obtendrá utilizando la definición dada al inicio de esta sección generalizada a intervalos significativos representados por Δ, e interpretando el fallo como parámetro y la ecuación del residuo como función de transferencia. Una primera definición de sensibilidad del residuo $r_i(k)$ respecto al fallo $f_j(k)$ será:

$$S_i^j(k) = \frac{\Delta r_i(k)}{\Delta f_j(k)} \frac{f_j(k)}{r_i(k)} \qquad (11.38)$$

El hecho de tomar incrementos de la forma $\Delta r_i(k)$ y $\Delta f_j(k)$ responde a ls situaciones reales, en la que dichos incrementos deben medirse experimentalmente y su carácter estocástico no permite establecer gran precisión. El inconveniente de esta definición es que la sensibilidad puede variar su valor en función del tamaño de los intervalos tomados, sin embargo la precisión para el diagnóstico es más cualitativa que cuantitativa.

Sin embargo, esta definición no es útil debido a su dificultad de cálculo, ya que nominalmente $r_i(k)$ y $f_i(k)$ son ambos nulo. Si se tiene en cuenta sólo la primera parte de la definición

$$S_i^j(k) = \frac{\Delta r_i(k)}{\Delta f_j(k)} \qquad (11.39)$$

Para el caso en que se diagnostican fallos en sensores y actuadores esta ecuación se traduce en

$$\frac{\Delta r_i(k)}{\Delta f_j(k)} = \hat{h}_i(z) \qquad (11.40)$$

$$\frac{\Delta r_i(k)}{\Delta f_j(k)} = \hat{g}_i^j(z) \qquad (11.41)$$

Para que estos polinomios en z represente un valor numérico de la sensibilidad es necesario adoptar un segundo criterio, que ser estático

$$\begin{aligned} S_i &= \hat{h}_i(z)\Big|_{z=1} \\ S_i^j &= \hat{g}_j^i\Big|_{z=1} \end{aligned} \qquad (11.42)$$

o dinámico

$$S_i = \max_{l \le m} \left\{ \sum_{k=0}^{l} \hat{h}_{ik}(z) \right\} \qquad (11.43)$$

$$S_i^j = \max_{l \le m} \left\{ \sum_{k=0}^{l} \hat{g}_j k^i(z) \right\} \qquad (11.44)$$

si bien este segundo criterio en la práctica no resulta útil, debido a que la necesaria utilización de filtros sobre los residuos oculta en general los picos que mostraría el efecto de la sensibilidad dinámica.

En la práctica esta definiciones resultan insuficientes al no tener en cuenta la existencia de ruido que oculta la información de la existencia de un fallo. Esto significa que las sensibilidades anteriores quedan condicionadas por la cantidad de ruido que aparece en los residuos habitualmente. Es decir, la desviación típica de los residuos o en su caso las envolventes establecerán los márgenes de confianza para la declaración del residuo como indicativo de la presencia de un fallo.

Sea σ_i la estimación de la desviación típica del residuo $r_i(k)$, en ausencia de fallo, entonces es conveniente definir la sensibilidad como

$$Sd_i^j = \frac{S_i^j}{\sigma_i} \qquad (11.45)$$

Esta sensibilidad está calculada a partir de un valor absoluto del fallo, pero en general la magnitud el fallo se mide de forma relativa al valor nominal de la variable a la que afecta, por lo que la sensibilidad relativa se debería definir como

$$Sdr_i^j = \frac{S_i^j}{\sigma_i} \bar{x}_j \qquad (11.46)$$

donde $\bar{x}_j$ representa una estimación del valor medio de la variable x_j asociada al fallo f_j, siempre y cuando

$$\left| x_j \right| \ge \varepsilon \qquad (11.47)$$

Si el valor medio de una variable es muy próximo a cero se puede tomar una estimación de su varianza $\sigma^2(x_j)$, entonces la definición de la sensibilidad relativa se debe reescribir de la siguiente manera

$$Sdr_i^j = \frac{S_i^j}{\sigma_i} \sigma^2(\bar{x}_j) \qquad (11.48)$$

La interpretación de esta sensibilidad de forma cuantitativa, a través de su valor absoluto, permite predecir la detectabilidad del fallo f_j mediante el residuo $r_i(k)$. Además, la interpretación del signo de dicha sensibilidad, permite deducir el sentido en el que se produce el fallo, a partir del conocimiento del residuo.

Las estimaciones propuesta para la medida de la sensibilidad de los residuos frente a los fallos están limitadas por la calidad del modelo, así como por el conocimiento que pueda obtenerse sobre los fallos del proceso. Las estimaciones iniciales, obtenidas a partir de los datos experimentales del sistema en ausencia de fallo, pueden ser mejoradas con los datos históricos de fallos que se recojan.

11.7 Vector de Fallos: Estimación del Tamaño del Fallo

Finalmente, se propone un método para estimar el tamaño del fallo a partir de la información que nos proporcionan los residuos y la información de la sensibilidad de cada una de las ecuaciones de paridad respecto a cada uno de los fallos. La estimación del tamaño del fallo tiene una doble finalidad, en primer

lugar aislar el fallo o fallos más significativos y en segundo lugar dar una medida de la importancia del mismo.

Para estimar el tamaño del fallo partiremos de la definición de sensibilidad

$$\Delta f_j(k) = \frac{\Delta r_i(k)}{S_i^j} \qquad (11.49)$$

o bien, análogamente:

$$\Delta f_j(k) = \frac{\Delta r_i(k)}{Sd_i^j}\frac{1}{\sigma_i} \qquad (11.50)$$

bien, finalmente

$$\frac{\Delta f_j(k)}{\bar{x}_j} = \frac{\Delta r_i(k)}{Sd_i^j}\frac{1}{\sigma_i} \qquad (11.51)$$

Es decir, si se considera que existen n_j elementos en la columna j-ésima de la matriz de sensibilidades

$$\Omega = \begin{bmatrix} Sd_1^1 & Sd_1^2 & \cdots & Sd_1^j & \cdots & Sd_1^n \\ Sd_2^1 & Sd_1^2 & \cdots & Sd_2^j & \cdots & Sd_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Sd_i^1 & Sd_i^2 & \cdots & Sd_i^j & \cdots & Sd_i^n \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ Sd_m^1 & Sd_m^2 & \cdots & Sd_m^j & \cdots & Sd_m^n \end{bmatrix} \qquad (11.52)$$

tales que $\left|Sd_i^j\right| > \varepsilon$, siendo ε un valor que limita la sensibilidad mínima que debe tenerse en cuenta para detectar la presencia de fallo, las expresiones proporcionadas anteriormente permiten realizar una estimación del tamaño del fallo según

$$\Delta f_j = \frac{1}{n_j}\sum_{i=1}^{m}\frac{\Delta r_i}{S_i^j} \qquad (11.53)$$

para todo S_i^j tal que $\left|Sd_i^j\right| > \varepsilon$.

Por otra parte, definiendo

$$W_i^j = \begin{cases} \dfrac{1}{n_j}\dfrac{1}{Sd_i^j}\dfrac{1}{\sigma_i} & \text{si } \left|Sd_i^j\right| > \varepsilon \\ 0 & \text{si } \left|Sd_i^j\right| \le \varepsilon \end{cases} \qquad (11.54)$$

o bien,

$$Wr_i^j = \begin{cases} \dfrac{1}{n_j}\dfrac{1}{Sdr_i^j}\dfrac{1}{\sigma_i} & \text{si } \left|Sd_i^j\right| > \varepsilon \\ 0 & \text{si } \left|Sd_i^j\right| \le \varepsilon \end{cases} \qquad (11.55)$$

se pueden definir las matrices

$$\Lambda = \begin{bmatrix} W_1^1 & W_1^2 & \cdots & W_1^j & \cdots & W_1^n \\ W_2^1 & W_1^2 & \cdots & W_2^j & \cdots & W_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ W_i^1 & W_i^2 & \cdots & W_i^j & \cdots & W_i^n \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ W_m^1 & W_m^2 & \cdots & W_m^j & \cdots & W_m^n \end{bmatrix} \quad (11.56)$$

y

$$\Lambda_r = \begin{bmatrix} Wr_1^1 & Wr_1^2 & \cdots & Wr_1^j & \cdots & Wr_1^n \\ Wr_2^1 & Wr_1^2 & \cdots & Wr_2^j & \cdots & Wr_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Wr_i^1 & Wr_i^2 & \cdots & Wr_i^j & \cdots & Wr_i^n \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ Wr_m^1 & Wr_m^2 & \cdots & Wr_m^j & \cdots & Wr_m^n \end{bmatrix} \quad (11.57)$$

donde se han considerado m ecuaciones de paridad y n posibles fallos. La columna j representa los pesos con los que los residuos son afectados por el fallo f_j. La fila i representa los pesos del residuo $r_i(k)$ frente a cada fallo.

Esta formulación permite realizar una estimación directa del **vector de tamaño de fallos** ΔF mediante

$$\Delta F = r^T(k)\Lambda \quad (11.58)$$

o bien, del tamaño relativo

$$\Delta F_r = r^T(k)\Lambda_r \quad (11.59)$$

11.8 Metodología de Diagnosis basada en el algoritmo DMP

La metodología de diagnosis de fallos Diagnostic Model Processor formulada por Petti [Petti92] parte se suponer que se dispone del modelo del proceso, formado por un conjunto de ecuaciones que describen su funcionamiento. Estas ecuaciones pueden ser desde simples ecuaciones que permiten determinar el rango de valores esperado para una determinada variable hasta ecuaciones más complicadas como, por ejemplo, ecuaciones diferenciales. Asociada con cada ecuación del modelo existen unos límites de tolerancia que indican cuando dicha ecuación ya no es representativa de la dinámica del proceso. Las ecuaciones del modelo se escriben en una forma tal que su valor en condiciones ideales debería ser cero. Pero la existencia de ruido, errores de modelo y fallos evitan que valgan cero. La discrepancia de cada ecuación respecto a cero se denomina residuo tal como hemos visto hasta ahora. Los límites de tolerancia son los valores inferior y superior esperados cuando el residuo está libre de fallos, dentro de los cuales la ecuación se considera que se cumple y por lo tanto, el residuo no es indicativo de la presencia de un fallo. Así mismo asociado con cada ecuación existen una serie de condiciones que si se cumple, se garantiza la satisfacción de la misma.

- **Vector de Residuos**

La metodología de diagnóstico empieza calculando un **vector de residuos** a partir de la ecuaciones del modelo en forma residual, e, a partir de los datos obtenidos a partir de medidas realizadas sobre el proceso real P. El residuo para la j-ésima ecuación del modelo dependerá, por lo tanto, de la ecuación

matemática de dicho residuo C_j, de las medidas realizadas P i del cumplimiento de las condiciones para la satisfacción de la misma, dicha dependencia se puede expresar de la siguiente manera

$$e_j = C_j(P, a) \qquad (11.60)$$

- **Vector de Satisfacción de las Ecuaciones Residuales: Toma de Decisión**

Puesto que los residuos obtenidos a partir de las ecuaciones del modelo no son uniformes en magnitud, se normalizan a valores comprendidas entre -1 y 1 indicando cual es el grado de satisfacción para cada ecuación: 0 indica que la ecuación es satisfecha a la perfección, 1 indica que la ecuación es violada severamente por exceso y -1 indica que la ecuación es severamente violada por defecto. El cálculo de dicho valores a partir de las ecuaciones residuales forman el **vector de satisfacción**, sf, que evalúa de forma fuzzy el grado de violación de cada residuo respecto al valor de tolerancia aceptable τ y proporciona un valor normalizado entre -1 y 1. La forma de calcular dicho vector de satisfacción es través de la función de pertenencia de Kramer, aplicada sobre el residuo y su tolerancia. Para la j-ésima ecuación del residuo,

$$sf_j = \frac{(e_j / \tau_j)^n}{1 + (e_j / \tau_j)^n} \qquad (11.61)$$

El valor de sf_j es positivo para residuos e_j positivos, mientras que es negativo para residuos e_j negativos. La función de Kramer es una función sigmoidal general con una pendiente que viene determina por la constante n. Si la tolerancia no es simétrica respecto al origen, se utiliza una tolerancia superior para residuos positivos y una de inferior para residuos negativos.

- **Matriz de Sensibilidades**

A partir de las ecuaciones residuales, se define una matriz de sensibilidad, S, que describe la relación de cada ecuación del modelo con sus condiciones de cumplimiento. Esta matriz se utilizará para ponderar los valores del vector de satisfacción sf, para obtener la evidencia de la existencia de un fallo. El j-ésimo elemento de la matriz S, que representa la sensibilidad de la j-ésima ecuación del modelo respecto a la i-ésima condición de cumplimiento se calcula como sigue

$$S_{ij} = \frac{\partial C_j / \partial a_i}{|\tau_j|} \qquad (11.62)$$

Cuanto mayor sea la derivada parcial de una ecuación respecto a una condición de cumplimiento, más sensible será dicha ecuación a detectar desviaciones de su cumplimiento. De forma similar, ecuaciones que posean una tolerancia τ elevada, será menos sensibles puesto que son más difíciles de violar. Las ecuaciones del modelo que son independientes de una condición de cumplimiento tienen asociada una sensibilidad para dicha condición igual a cero. Muchas ecuación del modelo pueden ser no lineales en algunas condiciones de cumplimiento. En ese caso las derivadas parciales se estiman mediante aproximación lineal. Así mismo, las condiciones de cumplimiento pueden ser:

1. **Implícitas** respecto a una ecuación (por ejemplo, para el caso de fallos en sensores y actuadores) en cuyo caso, la sensibilidad frente a la condición de cumplimiento vale 1, a menos que la experiencia demuestre lo contrario.
2. **Explícitas** respecto a una ecuación (por ejemplo, para el caso de fallos en el propio proceso) en cuyo caso, la sensibilidad se obtiene mediante la derivación parcial de la ecuación del modelo respecto la variación del parámetro del modelo.

- **Vector de Fallos**

Para obtener una conclusión sobre si se satisface una condición de cumplimiento, que normalmente, se asocia a un estado de fallo, se deben combinar las evidencias obtenidas a partir de las medidas realizadas sobre el proceso real y ponderadas a través del vector de satisfacción, sf, con su importancia relativa obtenida a través de la matriz de sensibilidad S. El método de combinación proporciona una medida normalizada del grado de cumplimiento de cada condición, a_i, que permite realizar contrastar las evidencias obtenidas a partir de las medidas realizadas sobre el modelo, que pueden indicar la presencia de fallos. El vector de fallos, F se define mediante

$$F_i = \frac{\sum_{j=1}^{N} (S_{ij} \cdot sf_j)}{\sum_{j=1}^{N} |S_{ij}|} \qquad (11.63)$$

donde N es el número de ecuaciones del modelo. Este método de combinación permite que los valores de sf que son más sensibles a desviaciones de una condición a_i sean tenidos en cuenta con un peso más importante en el cálculo de F_i. Un valor de una componente del vector de fallos, F_i, cerca de -1 ó 1 es indicativo de la presencia de un fallo debido a la violación de la condición a_i.

- **Puntos débiles de la metodología DMP**

La metodología DMP es muy adecuada para a partir de un conjunto de residuos estructurados de los cuales ya se conocen los umbrales de variación (tolerancias) determinar cual es la causa del fallo. La metodología, sin embargo, no incluye como generar los residuos estructurados ni como determinar los umbrales de variación. En este trabajo proponemos completar dicha metodología proponiendo un mecanismo para la generación de los residuos y un mecanismo para la generación de los umbrales.

11.9 Nueva Metodología de Detección y Diagnosis de Fallos: DMP mejorado

A continuación vamos a presentar una nueva metodología de diagnosis de fallos basada en el algoritmo Diagnostic Model Processor (DMP) que recoge muchas de las ideas expuestas ha ahora en este capítulo:

- Normalización de los residuos respecto a su posible rango de variaciones aceptable (tolerancia o desviación típica)
- Determinación de una matriz de sensibilidad de cada residuo frente a cada fallo
- Determinación de un vector de fallos combinando combinando la matriz de sensibilidad con los residuos.

Sin embargo, dicha metodología recoge algunas ideas que se han expuesto que son importantes y que no recoge la técnica básica basada en el análisis de los residuos estructurados:

- Definición de un vector de satisfacción de los residuos que evalúa en tiempo real mediante evaluación fuzzy el grado de significación de cada uno de ellos.
- Definición de un vector de fallos que proporciona cual es el fallo o fallos más significativos a base combinar la satisfacción de cada residuo con su sensibilidad.

Por otro lado, la metodología DMP no proporciona:

- Como determinar los residuos ni que número de residuos se debe escoger, ni tampoco una forma de evaluar con los residuos escogidos la propiedades de aislamiento que poseen.
- Tampoco proporciona la forma de determinar para cada instante de tiempo cual es el valor de tolerancia o desviación típica aceptable para residuo.

Teniendo en cuenta todo lo expuesto hasta el momento proponemos una nueva metodología de detección y diagnóstico de fallos robusta que tenga en cuenta todo lo que se ha expuesto en este capítulo y que resumimos a continuación:

- Generación de residuos estructurados a partir de ecuaciones de paridad teniendo en cuenta la propiedades de aislamiento a través del estudio de la matriz de incidencia.
- Generación del vector de satisfacción de cada residuo siguiendo la definición proporcionada por la metodología DMP, basada en una comparación difusa con el rango de tolerancia o variación posible de cada residuo
- Generación en tiempo real del rango de tolerancia o variación posible para cada residuo teniendo en cuenta los errores de modelado, el ruido y el punto de funcionamiento del proceso sobre el que se ha realizando la monitorización. La generación de dichos rangos de tolerancia para cada residuo se realizará mediante el algoritmo de generación de envolventes propuesto en esta tesis.
- Cálculo de la matriz de sensibilidad de cada residuo frente a los posibles fallos que pueden ser de tipo explícito (fallo en sensores y actuadores) o de tipo implícito (fallos en el propio proceso) utilizando la definición de sensibilidad dada por la metodología DMP. En la definición de sensibilidad dada por DMP aparece para cada sensibilidad su normalización respecto al rango de variación del residuo sobre el cual se está calculando. Dicha tolerancia es proporcionada también por el algoritmo de generación de envolventes.
- Cálculo del vector de fallos utilizando para la determinación del primero la definición dada por DMP, combinando el vector de satisfacción de los residuos con la matriz de sensibilidad utilizando una media aritmética ponderada.
- Estimación del tamaño del fallo utilizando la forma de cálculo propuesta por la metodología de análisis de residuos estructurados.

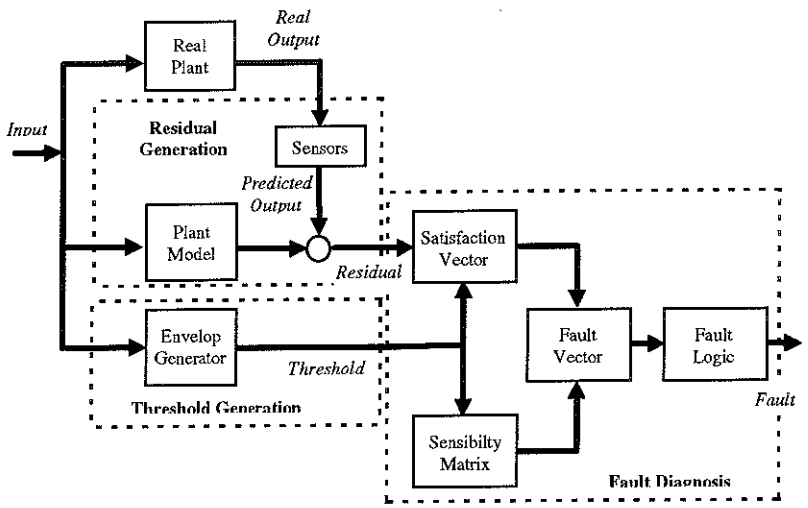


Fig. 11.8 Diagrama de Bloques del Sistema de Detección y Diagnóstico de Fallo basado en DMP mejorado

11.10 Ejemplo de Aplicación del DMP mejorado

En este apartado presentaremos los resultados obtenidos al aplicar el algoritmo de detección y diagnóstico de fallos DMP mejorado sobre un proceso de laboratorio basado en dos tanques de agua interconectados.

El proceso funciona de la siguiente manera: una bomba envía agua al primer depósito desde un depósito general. Desde este depósito el agua cae libremente a un segundo depósito a través una tubería, desde donde vuelve al depósito general a través de otra tubería. Este sistema puede ser modelado matemáticamente utilizando las leyes de la hidráulica.

Para poder medir las variables significativas del proceso de cara a poder monitorizar su correcto funcionamiento se han instalado en la planta cinco sensores que permiten medir: el caudal de la bomba (Q_b), el nivel del tanque 1 (h_1), el caudal de la tubería 1-2 (Q_{12}), el nivel del tanque 2 (h_2) y el caudal de salida del tanque 2 (Q_2).

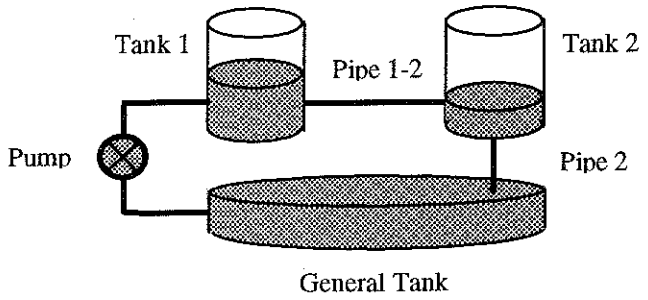


Fig. 11.9 El sistema de dos depósitos interconectados

11.10.1 Modelo del proceso

En primer lugar, vamos a deducir las ecuaciones del modelo del proceso:

- *Ecuación del modelo de la bomba*

$$Q_b(k) = aU_b(k) + b$$

donde $\mathbf{Q_b}$ es el caudal de la bomba, $\mathbf{U_p}$ es la tensión de la bomba y (**a**,**b**) son los parámetros del modelo de la bomba.

- *Ecuación del modelo de la tubería que une el depósito 1 y 2*

$$Q_{12}(k) = \left[h_1(k) - h_2(k)\right] * A + B$$

donde $\mathbf{Q_{12}}$ es el caudal de la tubería 1-2, $\mathbf{h_1}$ es el nivel del depósito 1, $\mathbf{h_2}$ es el nivel del depósito 2 y (**A**, **B**) son los parámetros del modelo de la tubería.

- *Ecuación del modelo de la tubería que une el depósito 2 y el general*

$$Q_2(k) = h_2(k) * C + D$$

donde $\mathbf{Q_2}$ es el caudal de la tubería 2, $\mathbf{h_2}$ es el nivel del depósito 2 y (**C**, **D**) son los parámetros del modelo de la tubería.

- *Ecuación del modelo del depósito 1*

$$h_1(k+1) = \frac{T}{S_1}\left[Q_p(k) - Q_{12}(k)\right] + h_1(k)$$

donde $\mathbf{Q_{12}}$ es el caudal de la tubería 1-2, $\mathbf{h_1}$ es el nivel del depósito 1, $\mathbf{Q_p}$ es el caudal de la bomba, $\mathbf{S_1}$ es el área de la sección del depósito 1 y **T** es el periodo de muestreo.

- *Ecuación del modelo del depósito 2*

$$h_2(k+1) = \frac{T}{S_2}\left[Q_{12}(k) - Q_2(k)\right] + h_2(k)$$

donde $\mathbf{Q_{12}}$ es el caudal de la tubería 1-2, $\mathbf{h_2}$ es el nivel del depósito 2, $\mathbf{Q_2}$ es el caudal de la tubería 2, $\mathbf{S_2}$ es el área de la sección del depósito 2 y **T** es el periodo de muestreo.

- *Modelo del sistema*

A partir de la aplicación de dichas leyes al sistema de depósitos se puede deducir el modelo del sistema completo en el espacio de estado, obteniéndose un sistema de segundo orden

$$\begin{bmatrix} h_1(k+1) \\ h_2(k+1) \end{bmatrix} = A \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} + Bu(k)$$

donde $h_1(k)$ y $h_2(k)$ son los niveles de los depósitos y u(k) es la tensión en la bomba. Utilizando las técnicas de identificación de sistemas podemos determinar las cotas para los parámetros del sistema

$$A_{max} = \begin{bmatrix} -0.0427 & 0.0523 \\ 0.0523 & -0.0564 \end{bmatrix} \text{ y } A_{min} = \begin{bmatrix} -0.0459 & 0.0485 \\ 0.0485 & -0.0608 \end{bmatrix}$$

$$B_{max} = \begin{bmatrix} 0.0131 & 0.0497 \\ 0 & -0.0536 \end{bmatrix} \text{ y } B_{min} = \begin{bmatrix} 0.0131 & 0.0461 \\ 0 & -0.0578 \end{bmatrix}$$

11.10.2 Generación de las envolventes utilizando el nuevo algoritmo

Si proporcionamos las matrices de parámetros del modelo sistema de depósitos interconectados obtenidas en el apartado anterior y le proporcionamos la misma entrada que le estamos aplicando al sistema real, el software de generación de envolventes que hemos desarrollado proporciona los siguientes resultados:

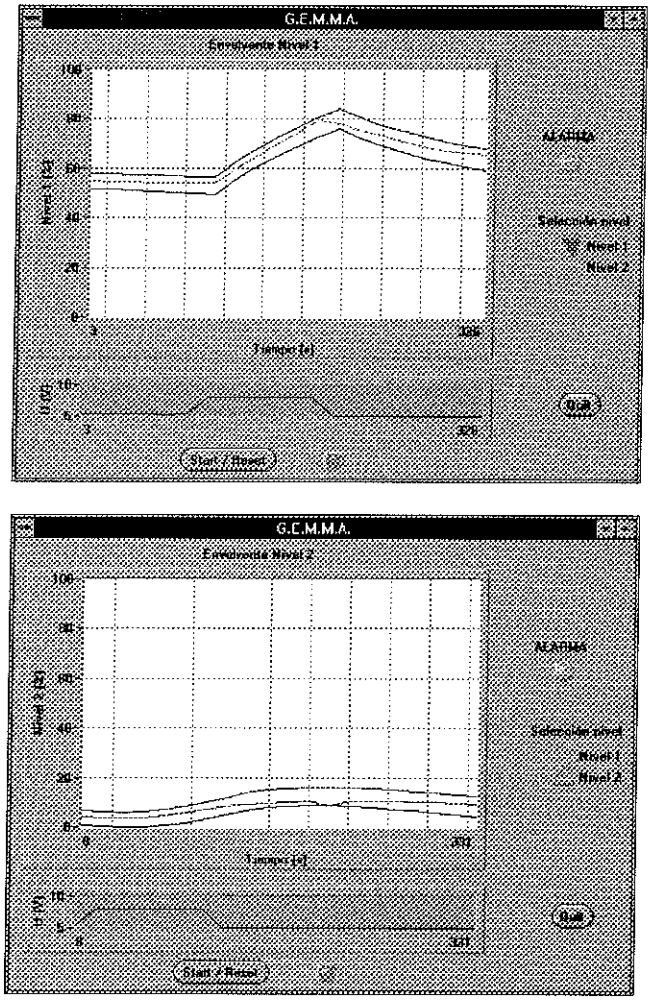


Fig. 11.10 Generación de Envolventes para el sistema de depósitos interconectados

11.10.3 Aplicación del algoritmo de diagnóstico DMP

A partir del modelo del proceso obtenido en el punto 11.10.1, vamos a aplicar la metodología de diagnóstico DMP.

- *Residuos, Tolerancias, Suposiciones y Vector de Satisfacción*

En primer lugar a partir de las ecuaciones del modelo, obtendremos las ecuaciones residuales de paridad utilizando su forma explícita o ARMA. Cada unas de estas ecuaciones de paridad lleva asociados unas suposiciones que si no se cumplen el residuo violará la tolerancia permitida, o sea, el umbral a partir del cual se considera indicativo de fallo. Utilizando las envolventes generadas a partir del nuevo algoritmo de generación de envolventes se obtendrán las tolerancias para cada una de las ecuaciones residuales. Finalmente a partir de las tolerancias y las ecuaciones residuales se definirán las componentes del vector de satisfacción, que tal como hemos dicho lo que nos indica es si la violación de un determinado residuo es no significativa comparándolo con su tolerancia o umbral, y utilizando como lógica de decisión la lógica fuzzy implementada aquí mediante una función sigmoidal. En la tabla 1 se presentan las ecuaciones de paridad, las componentes del vector de satisfacción y las suposiciones asociadas a cada ecuación de paridad.

Ecuaciones de Paridad	Vector de Satisfacción	Suposiciones
$e_1 = Q_p - \hat{Q}_p$	$sf_1 = \frac{(e_1/\tau_1)^4}{1+(e_1/\tau_1)^4}$	El sensor de caudal está OK (a_1). El parámetro **a** de la bomba está OK (a_2). El parámetro **b** de la bomba está OK (a_3).
$e_2 = Q_{12} - \hat{Q}_{12}$	$sf_2 = \frac{(e_2/\tau_2)^4}{1+(e_2/\tau_2)^4}$	El sensor de caudal de la tubería 1-2 está OK (a_4). El parámetro **A** de la tubería 1-2 está OK (a_5). El parámetro **B** de la tubería 1-2 está OK. (a_6).
$e_3 = Q_2 - \hat{Q}_2$	$sf_3 = \frac{(e_3/\tau_3)^4}{1+(e_3/\tau_3)^4}$	El sensor de caudal de la tubería 2 está OK (a_7). El parámetro **C** de la tubería 2 está OK (a_8). El parámetro **D** de la tubería 2 está OK (a_9).
$e_4 = h_1 - \hat{h}_1$	$sf_4 = \frac{(e_4/\tau_4)^4}{1+(e_4/\tau_4)^4}$	El sensor de nivel del tanque 1 está OK (a_{10}). La sección del tanque 1 está OK (a_{11}).
$e_5 = h_2 - \hat{h}_2$	$sf_5 = \frac{(e_5/\tau_5)^4}{1+(e_5/\tau_5)^4}$	El sensor de nivel del tanque2 está OK (a_{12}). La sección del tanque 1 está OK (a_{13}).

Tabla 1. Residuos, Vector de Satisfacción y Suposiciones para el sistema de dos tanques interconectados

- *Matriz de Sensibilidades*

El siguiente paso en la metodología DMP consiste en calcular la matriz de sensibilidad. Cada uno de los componentes de dicha matriz mide el grado de sensibilidad de cada ecuación de paridad respecto a la violación de cada uno de los residuos. En la tabla 2 se presenta la matriz de sensibilidad para el sistema de dos tanques interconectados.

	$\mathbf{e_1}$	$\mathbf{e_2}$	$\mathbf{e_3}$	$\mathbf{e_4}$	$\mathbf{e_5}$
$\mathbf{a_1}$	$S_{11} = \frac{1}{\lvert\tau_1\rvert}$	$S_{12} = 0$	$S_{13} = 0$	$S_{14} = 0$	$S_{15} = 0$
$\mathbf{a_2}$	$S_{21} = \frac{-U_p}{\lvert\tau_1\rvert}$	$S_{22} = 0$	$S_{23} = 0$	$S_{24} = \frac{-TU_p}{\lvert\tau_4\rvert S_1}$	$S_{25} = 0$
$\mathbf{a_3}$	$S_{31} = \frac{1}{\lvert\tau_1\rvert}$	$S_{32} = 0$	$S_{33} = 0$	$S_{43} = \frac{T}{\lvert\tau_4\rvert S_1}$	$S_{35} = 0$
$\mathbf{a_4}$	$S_{41} = 0$	$S_{42} = \frac{1}{\lvert\tau_2\rvert}$	$S_{43} = 0$	$S_{44} = 0$	$S_{45} = 0$
$\mathbf{a_5}$	$S_{51} = 0$	$S_{52} = \frac{-[h_1 - h_2]}{\lvert\tau_2\rvert}$	$S_{53} = 0$	$S_{54} = \frac{-T[h_1 - h_2]}{\lvert\tau_4\rvert S_1}$	$S_{55} = \frac{T[h_1 - h_2]}{\lvert\tau_5\rvert S_2}$
$\mathbf{a_6}$	$S_{61} = 0$	$S_{62} = \frac{-1}{\lvert\tau_2\rvert}$	$S_{63} = 0$	$S_{64} = \frac{T}{\lvert\tau_4\rvert S_1}$	$S_{65} = \frac{-T}{\lvert\tau_5\rvert S_2}$
$\mathbf{a_7}$	$S_{71} = 0$	$S_{72} = 0$	$S_{73} = \frac{1}{\lvert\tau_3\rvert}$	$S_{74} = 0$	$S_{75} = 0$
$\mathbf{a_8}$	$S_{81} = 0$	$S_{82} = 0$	$S_{83} = \frac{-h_2}{\lvert\tau_3\rvert}$	$S_{84} = 0$	$S_{85} = \frac{Th_2}{\lvert\tau_5\rvert S_2}$
$\mathbf{a_9}$	$S_{91} = 0$	$S_{92} = 0$	$S_{93} = \frac{-1}{\lvert\tau_3\rvert}$	$S_{94} = 0$	$S_{95} = \frac{T}{\lvert\tau_5\rvert S_2}$
$\mathbf{a_{10}}$	$S_{101} = 0$	$S_{102} = 0$	$S_{103} = 0$	$S_{104} = \frac{1}{\lvert\tau_4\rvert}$	$S_{105} = 0$
$\mathbf{a_{11}}$	$S_{111} = 0$	$S_{112} = 0$	$S_{113} = 0$	$S_{114} = \frac{-T[Q_p - Q_{12}]}{\lvert\tau_4\rvert S_1^2}$	$S_{115} = 0$
$\mathbf{a_{12}}$	$S_{121} = 0$	$S_{122} = 0$	$S_{123} = 0$	$S_{124} = 0$	$S_{125} = \frac{1}{\lvert\tau_5\rvert}$
$\mathbf{a_{13}}$	$S_{131} = 0$	$S_{132} = 0$	$S_{133} = 0$	$S_{134} = 0$	$S_{135} = \frac{T[Q_{12} - Q_2}{\lvert\tau_5\rvert S_1^2}$

Tabla 2. Matriz de Sensibilidades

- *Vector de Fallos*

Una vez definido el vector de satisfacción y la matriz de sensibilidades, ambos deben combinarse junto con la tolerancia obtenida a partir del algoritmo de generación de envolventes en el vector de fallos, que relaciona cada violación de una ecuación de paridad con la violación de la suposición asociada. En la tabla 3 se muestran las componentes del vector de fallos para el siguiente de los dos tanques interconectados.

Suposición	Vector de Fallos
$\mathbf{a_1}$	$F_1 = sf_1$
$\mathbf{a_2}$	$F_2 = \dfrac{-\dfrac{sf_1}{\tau_1} - \dfrac{Tsf_4}{S_1\tau_4}}{\dfrac{1}{\tau_1} + \dfrac{T}{S_1\tau_4}}$
$\mathbf{a_3}$	$F_3 = \dfrac{\dfrac{sf_1}{\tau_1} + \dfrac{Tsf_4}{S_1\tau_4}}{\dfrac{1}{\tau_1} + \dfrac{T}{S_1\tau_4}}$
$\mathbf{a_4}$	$F_4 = sf_2$
$\mathbf{a_5}$	$F_5 = \dfrac{-\dfrac{sf_2}{\tau_2} + \dfrac{Tsf_4}{S_1\tau_4} - \dfrac{Tsf_5}{S_2\tau_5}}{\dfrac{1}{\tau_2} + \dfrac{T}{S_1\tau_4} + \dfrac{T}{S_2\tau_5}}$
$\mathbf{a_6}$	$F_6 = \dfrac{-\dfrac{sf_2}{\tau_2} + \dfrac{Tsf_4}{S_1\tau_4} - \dfrac{Tsf_5}{S_2\tau_5}}{\dfrac{1}{\tau_2} + \dfrac{T}{S_1\tau_4} + \dfrac{T}{S_2\tau_5}}$
$\mathbf{a_7}$	$F_7 = sf_3$
$\mathbf{a_8}$	$F_8 = \dfrac{-\dfrac{sf_3}{\tau_3} + \dfrac{Tsf_5}{S_2\tau_5}}{\dfrac{1}{\tau_3} + \dfrac{T}{\tau_5}}$
$\mathbf{a_9}$	$F_9 = \dfrac{-\dfrac{sf_3}{\tau_3} + \dfrac{Tsf_5}{S_2\tau_5}}{\dfrac{1}{\tau_3} + \dfrac{T}{\tau_5}}$
$\mathbf{a_{10}}$	$F_{10} = sf_4$
$\mathbf{a_{11}}$	$F_{11} = sf_4$
$\mathbf{a_{12}}$	$F_{12} = sf_5$
$\mathbf{a_{13}}$	$F_{13} = sf_5$

Tabla 3. Vector de Fallos

- *Lógica de Diagnóstico*

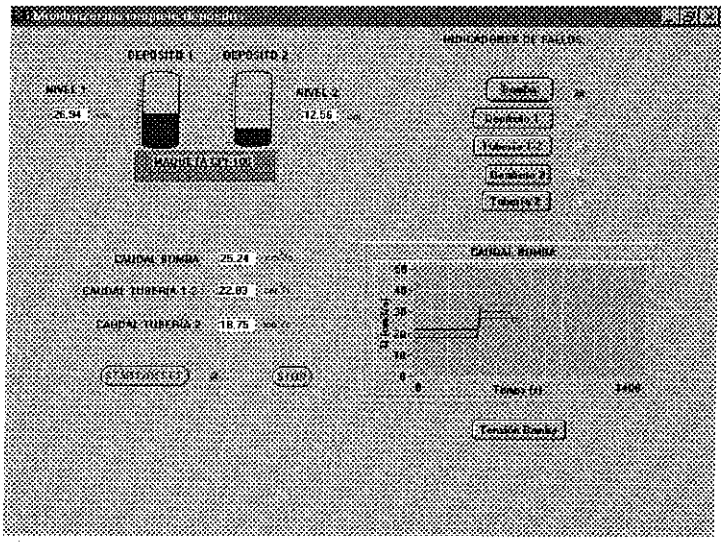
El último paso de la metodología DMP consiste en implementar una lógica que permita mapear cada uno de los fallos en los componentes a fallos en el proceso. Los fallos posibles que deseamos identificar son los siguientes:

- Fallo en la Bomba
- Fallo en la Tuberia 1-2
- Fallo en Tuberia 2
- Fallo en el Depósito 1.
- Fallo en el Depósito 2.

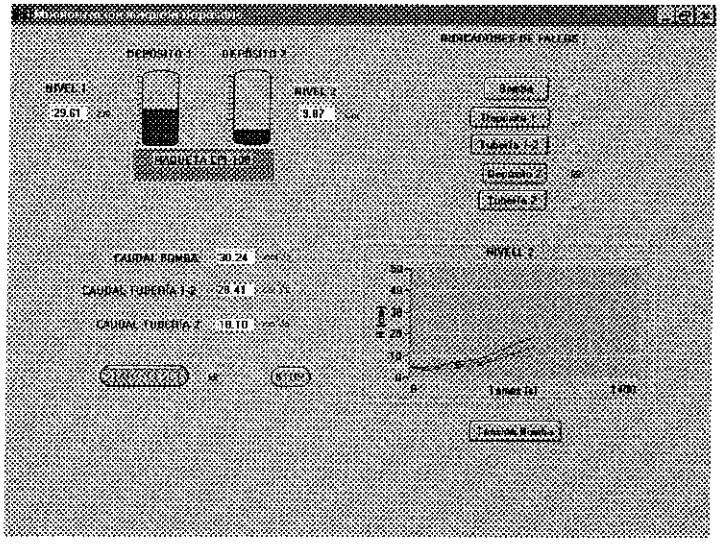
En nuestro ejemplo esta lógica se implementa mediante una tabla de la verdad, que mapea los valores del vector de fallos sobre cada uno de los posibles fallos que se desean identificar.

11.10.4 Resultados experimentales de la implementación Sistema de Diagnóstico

A continuación se presentan una aplicación del sistema de diagnóstico robusto presentado anterior aplicado sobre el proceso industrial de laboratorio basado en los dos tanques interconectados que se ha descrito en la sección 3 de este trabajo. Para poder diagnosticar correctamente los fallos de la planta se han instalado cinco sensores que miden: caudal de la bomba (Q_b), nivel del depósito 1 (h_1), caudal de la tubería 1-2 (Q_{12}), nivel del depósito 2 (h_2) y caudal de la tubería 2 (Q_2). La implementación del sistema de diagnóstico se ha realizado utilizando el entorno LabWindows/CVI de National Instruments. En la Fig.6 y Fig.7 se presentan algunos resultados de dicha implementación real, donde se muestra la detección de dos fallos graduales: uno sobre la bomba y otros sobre el depósito 2.



(a)



(b)

Fig. 11.11 (a) Fallo Gradual en la Bomba y
(b) Fallo Gradual en el Depósito 2

11.11 Inclusión de medidas de sensores en el algoritmo de generación de envolventes

Una vez presentada la nueva metodología de detección y diagnóstico de fallos a continuación vamos a presentar como el algoritmo de generación de envolventes puede incluir o tener en cuenta las medidas de los sensores.

Para ello, en primer lugar vamos a recordar como funciona un estimador de estado.

11.1.1 Estimación d Estado

- **Estimadores de orden total**

Un método para calcular todos los estados de un sistema dinámico que se pueden considerar es a partir del modelo de la dinámica de la planta en el espacio de estado

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) \qquad (11.64)$$

donde $\hat{x}$ es la estimación de los estados reales. Si se conoce el valor de los parámetros del modelo A y B y la entrada al sistema u(t), entonces la estimación de los estados del sistema será correcta siempre que se conozca el estado inicial del sistema x(0), de forma que podamos hacer igualarla a $\hat{x}(0)$:

$$\hat{x}(0) = x(0) \qquad (11.65)$$

En la Fig. 11.12 se muestra la estructura de este estimador de estado, que a partir de ahora denominaremos de "lazo abierto".

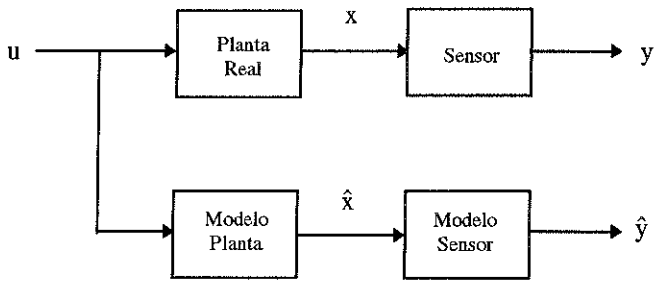


Fig. 11.12 Estimador de Estado en Lazo Abierto

Sin embargo, es precisamente la falta de información acerca de x(0) lo que requiere la construcción de un estimador. De otra forma, los estados estimados seguirían exactamente los estados verdaderos. De ahí que si se desconoce cual es el estado inicial real del sistema y se toma igual a cero su estimación, los estados estimados presentarían un error continuamente creciente, o bien, un error que tendería muy lentamente a cero. Además, los errores en la estimación debido a la incertidumbre asociada al modelo harían que el cálculo divergiera muy lentamente del estado real.

Para estudiar la dinámica de este estimador de estado se define el error de estimación como:

$$e_x(t) = x(t) - \hat{x}(t) \qquad (11.66)$$

Por lo tanto, la dinámica de este error en el sistema vendrá dada por:

$$\begin{aligned} \dot{e}_x(t) &= Fe_x(t) \\ e_x(0) &= x(0) - \hat{x}(0) \end{aligned} \qquad (11.67)$$

El error converge a cero para un sistema estable, A estable, pero no existe la posibilidad de influir en la velocidad de la convergencia del estado estimado hacia es estado verdadero. Además, el error está convergiendo a cero a la misma velocidad que la dinámica natural de sistema, o sea, de A. Si dicha velocidad de convergencia fuera satisfactoria, no se requeriría realizar ninguna corrección de la estimación a partir de las medidas obtenidas a partir de alguno de los sensores colocados en la planta. Si por el contrario, la velocidad de convergencia no es la adecuada, se debería utilizar dicha corrección. Para ello deberíamos recurrir a un esquema de estimador como en el que se muestra en la Fig. 11.13 en el que se corrige continuamente la predicción realizada por el modelo del sistema mediante la diferencia entre la salida medida por un sensor colocado en la planta y la estimación que nos proporciona el modelo. Tal como se puede observar en la Fig. 11.13 la ecuación que describe la dinámica de este estimador, que denominaremos estimador de "lazo cerrado" viene dada por

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L(y(t) - C\hat{x}(t)) \qquad (11.68)$$

donde L es una ganancia proporcional

$$L = [l_1, l_2, \cdots . l_n]^T \qquad (11.69)$$

que se elige a fin de alcanzar unas características de error satisfactorias.

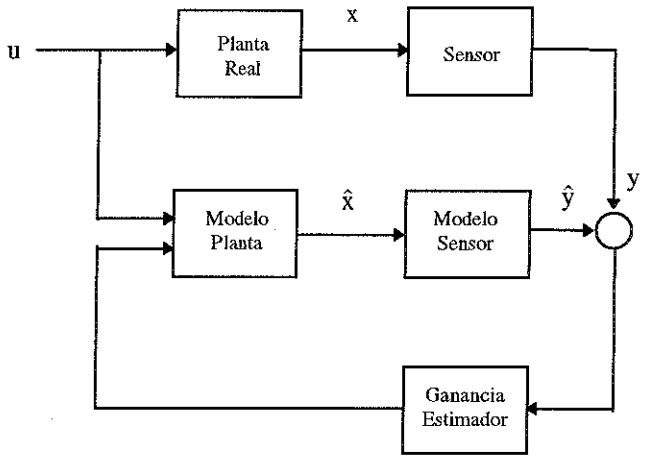


Fig. 11.13 Estimador de Estado en Lazo Cerrado

La dinámica del error se puede obtener mediante

$$e_x(t) = (A - LC)e_x(t) + G_1 w(t) \qquad (11.70)$$

La ecuación característica del error vendrá dada por

$$\det[sI - (A - LC)] = 0 \qquad (11.71)$$

Si se elige L de forma que A-LC tenga raíces razonablemente rápidas y estables, entonces $e_x(t)$ decaerá a cero mientras no exista una entrada perturbante w(t), independiente de $e_x(0)$, pudiéndose además elegir que dicha dinámica sea más rápida que la dinámica del sistema en lazo abierto A.

Se puede observar que la ecuación de la dinámica del error está forzada solamente por el término w(t). Esto es consecuencia de la suposición de que A, B y C son idénticas en la planta y en el estimador. Por lo tanto, para w(t)=0, el error calculado $e_x(t)$ convergirá a cero y se mantendrá en dicho valor, independientemente del entrada u(t). En cambio, si no se tiene un modelo muy preciso de la planta, es decir, existe incertidumbre asociada con A, B y C, la dinámica del error ya no estará gobernada por la ecuación anterior. Sin embargo, generalmente se podrá elegir L de manera que la dinámica del error sea

estable y su valor de dicho error sea razonablemente pequeño, incluso con errores de modelado y entrada perturbantes.

- **Estimadores de orden reducido**

En el caso de los estimadores de orden total, el vector de estado se reconstruye totalmente empleando la medición de alguno de los estados mediante un sensor. Si los sensores no tuvieran ruido, parecería redundante el hecho de estimar también los estados que se están midiendo directamente, la pregunta es como reducir la complejidad del estimador completo empleando los estados que se miden directa y exactamente. La respuesta es utilizar como veremos en este apartado utilizar un estimador de orden reducido. Sin embargo, en la práctica, debido a la existencia de ruido en las medidas proporcionadas por los sensores, resulta mejor implantar un estimador de orden total, debido a que el estimador filtra el ruido asociado con las mediciones a parte de estimar los estados que no son conocidos.

El estimador de orden reducido disminuye el número de integradores requeridos en el número de salidas medidas. Para determinar este estimador, se divide el vector de estado en dos partes: una parte x_a, que se mide directamente (x_a=y), y otra x_b, que representa los estados restantes que hay que calcular. Si a las matrices del sistema se les hacen las particiones correspondientes, la descripción completa de dicho sistema viene dada por

$$\begin{bmatrix}\dot{x}_a\\ \dot{x}_b\end{bmatrix}=\begin{bmatrix}A_{aa} & A_{ab}\\ A_{ba} & A_{bb}\end{bmatrix}\begin{bmatrix}x_a\\ x_b\end{bmatrix}+\begin{bmatrix}B_a\\ B_b\end{bmatrix}u \qquad (11.72)$$

$$y=\begin{bmatrix}I & 0\end{bmatrix}\begin{bmatrix}x_a\\ x_b\end{bmatrix}$$

La dinámica de los estados no medidos viene dada por

$$\dot{x}_b = A_{bb}x_b + \underbrace{A_{ba}x_a + B_b u}_{\text{Entrada conocida}} \qquad (11.73)$$

donde los dos últimos términos de la derecha se conocen y pueden considerarse como la entrada en la dinámica x_b. Como x_a=y, y la dinámica de los estados medidos viene dada por la ecuación escalar

$$\dot{x}_a = \dot{y} = A_{aa}y + A_{ab}x_b + B_a u \qquad (11.74)$$

Si se agrupan los dos términos conocidos en la ecuación anterior

$$\underbrace{\dot{y} - A_{aa}y - B_a u}_{\text{Medida conocida}} = A_{ab}x_b \qquad (11.75)$$

se obtiene una relación entre cantidades conocidas en el lado izquierdo, consideradas como mediciones, y los estados desconocidos en el lado derecho. Por tanto, las ecuaciones anteriores tienen la misma relación con el estado x_b que la ecuación original con el estado x. Siguiendo esta línea de razonamiento, se pueden establecer las siguientes sustituciones en las ecuaciones del estimador original para obtener un estimador (de orden reducido) de los estados no medidos x_b

$$x \leftarrow x_b$$

$$A \leftarrow A_{bb}$$

$$Bu \leftarrow A_{ba}y + B_b u$$

$$y \leftarrow \dot{y} - A_{aa}y - B_a u$$

$$C \leftarrow A_{ab}$$

Por tanto, las ecuaciones del estimador de orden reducido se obtienen realizando las sustituciones anteriores en el estimador de orden total

$$\dot{\hat{x}}_b = A_{bb}\hat{x}_b + \underbrace{A_{ba}y + B_b u}_{\text{Entrada}} + L(\underbrace{\dot{y} - A_{aa}y - B_a u}_{\text{Medida}} - A_{ab}\hat{x}_b) \qquad (11.76)$$

Si se define el error del estimador como

$$\tilde{x}_b = x_b - \hat{x}_b \qquad (11.77)$$

entonces la dinámica del error vendrá dada por

$$\dot{\tilde{x}} = (A_{bb} - LA_{ab})\tilde{x}_b \qquad (11.78)$$

y su ecuación característica vendrá dada por

$$\det\left[sI - \left(A_{bb} - LA_{ab}\right)\right] = 0 \qquad (11.79)$$

Finalmente, se pude reescribir la ecuación que nos proporciona la estimación de los estados no medidos como

$$\dot{\hat{x}}_b = (A_{bb} - LA_{ab})\hat{x}_b + (A_{ba} - LA_{aa})y + (B_b - LB_a)u + L\dot{y} \qquad (11.80)$$

En dicha ecuación aparece la derivada de las medidas del sensor. Dicha derivación, en general, amplificará el ruido, de manera su empleo resulta inaceptable. Para evitar que aparezca la derivada de las medidas del sensor en la ecuación del estimador, se introduce un nuevo estado

$$x_c = \hat{x}_b - Ly \qquad (11.81)$$

de forma, que en función de este nuevo estado, la expresión del estimador de orden reducido vendrá dada por

$$\dot{x}_c = (A_{bb} - LA_{ab})\hat{x}_b + (A_{ba} - LA_{aa})y + (B_b - LB_a)u \qquad (11.82)$$

en la que la derivada de las mediciones no aparece directamente.

11.1.2 Inclusión de medidas de sensores en el algoritmo de generación de envolventes

El algoritmo de generación de umbrales adaptativos presentado en esta tesis se ha formulado hasta el momento para verificar una medida obtenida mediante un sensor de una de las variables del sistema real confrontándola con la predicción de dicha medida obtenida mediante el algoritmo de generación de envolventes a partir de un modelo del sistema con una incertidumbre en sus parámetros conocida y partir de la entrada del sistema. Tal como está formulado dicho algoritmo sería capaz de monitorizar y detectar fallos en sistemas con un único sensor. Pero si se dispone de más de un sensor, puede resultar ventajoso incluir la información adicional proveniente del resto de sensores en el algoritmo de generación de envolventes. De forma que el algoritmo de generación de envolventes podrá contrastar la medida de un sensor frente a la predicción proporcionada por el modelo con incertidumbre en los parámetros conocida, las entradas y la información proporcionada por el resto de sensores. La información adicional proporcionada por el resto de sensores reducirá el tamaño del intervalo dentro del cual se debe encontrar la medida del sensor monitorizado para que el algoritmo de detección no indique que se ha producido un fallo. La reducción del tamaño de los intervalos de validez de las medidas obtenidas por el sensor monitorizado, permitirá la detección de fallos de menor tamaño.

Durante el proceso de maximización/minimización para la obtención del intervalo de validez de la medida del sensor monitorizado, el algoritmo de generación de envolventes selecciona el valor de parámetros del modelo que determina la historia temporal de todos los estados del sistema. Si se dispone de medidas adicionales, pero dichas medidas no son consideradas por el algoritmo de generación de envolventes, existirán discrepancias entre los estados calculados mediante dicho algoritmo y las medidas obtenidas mediante los sensores. El algoritmo supondrá que se produce la peor variación posible de los parámetros, de forma que predecirá la peor historia temporal posible para los estados del sistema. Por otro lado, las medidas obtenidas de los sensores, reflejan el hecho de que la mayoría de las veces no se producen ni la peor variación de los parámetros ni la peor historia temporal posible de los estados. Si

una de las medidas adicionales disponibles es un estado del sistema, el problema presentado hasta este momento puede ser resuelto eliminando el estado y las correspondiente fila y columna de la matriz del sistema, y añadiendo un nuevo término de entrada definido como el producto de la columna eliminada y el estado medido. El algoritmo de generación de envolventes puede entonces aplicarse al sistema reducido así obtenido, simple que dicho sistema sea estable.

En general, sin embargo, no todas las medidas disponibles se corresponden a estados, y a menudo el sistema reducido es inestable. Para estos casos, se puede utilizar una versión del algoritmo de generación de envolventes que utilice una estructura de tipo observador que tenga en cuenta las medidas adicionales y que se pueda aplicar a todo tipo de sistemas, tanto los que al construir el sistema reducido sean estables como inestables. Dicha versión del algoritmo de generación de envolventes utiliza el siguiente modelo del sistema como punto de partida

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + K[z(k) - y(k)] \\ y(k) &= C_1 x(k) \end{aligned} \qquad (11.83)$$

donde la matriz C_1 incluye solo aquellas filas de la matriz C que corresponden a las medidas adicionales, y z(k) es el vector de medidas proporcionadas por el sensor situada en la planta correspondientes a las medidas obtenidas mediante el modelo y(k). La matriz de ganancia K tiene la misión de asegurar que las salidas del sistema predichas por el modelo tomarán valores próximos a las medidas obtenidas por los sensores durante el proceso de optimización, evitando que el algoritmo de optimización utilice valores de estados que no estén de acuerdo con las medidas. Al seleccionar la matriz de ganancia K se debe de tener en cuenta que se consiga la máxima proximidad posible entre el valor de las medidas obtenidas mediante los sensores y la estimación realizada por el modelo, dejando en un segundo plano cual es la posición de los polos del sistema en lazo cerrado.

11.12 Alternativas para el aislamiento de fallos en sensores

En la mayoría de aplicaciones industriales, el sistema de detección y aislamiento de fallos debe de monitorizar diversos sensores de forma simultánea. El algoritmo de generación de envolventes presentado en esta tesis es un bloque constructivo muy potente para la detección de fallos en sistemas dinámicos con errores de modelado en los parámetros conocidos.

El sistema de detección y aislamiento de fallos multisensor basado en el algoritmo presentado en este trabajo más simple utiliza un algoritmo de generación de envolventes para cada sensor tal como se ha presentado inicialmente, es decir, sin utilizar ninguna información adicional sobre las medidas obtenidas por otros sensores. Dicho sistema de detección y aislamiento de fallos proporciona cobertura a todos los sensores y puede manejar múltiples fallos en varios sensores sin ningún procesado adicional. El estado de cada sensor es determinado por su correspondiente algoritmo de generación de envolventes. La desventaja de este enfoque es su conservadurismo debido a que no tiene en cuenta las otras medidas disponibles, tal como se ha explicado en la sección anterior.

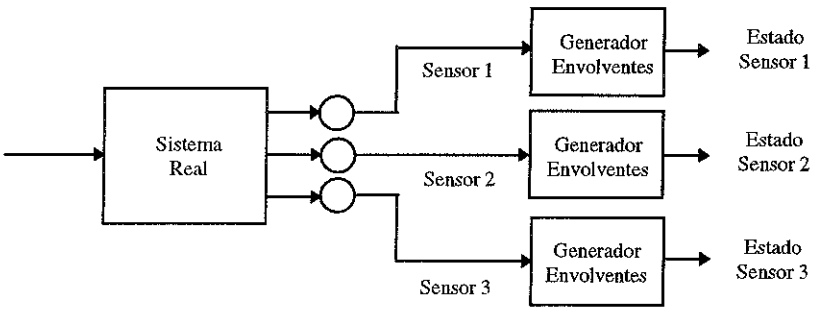


Fig. 11.14 Sistema de Detección y Aislamiento de Fallos de múltiples sensores mediante el algoritmo de generación de envolventes simple

Un sistema de detección y aislamiento de fallos más complejo utiliza un algoritmo de generación de envolventes que aprovecha la información adicional disponible proporcionada por el resto de sensores. Dicho sistema proporciona unos intervalos para la medidas posibles obtenidos a partir del algoritmo de generación de envolventes menores, y por lo tanto, puede detectar fallos menores que el sistema que no tiene en cuenta las otras medidas. Sin embargo, la utilización de medidas adicionales abre la posibilidad de tomar una decisión de cual es la causa del fallo, debido a se esta utilizado información proveniente de otros sensores para monitorizar el estado del sensor.

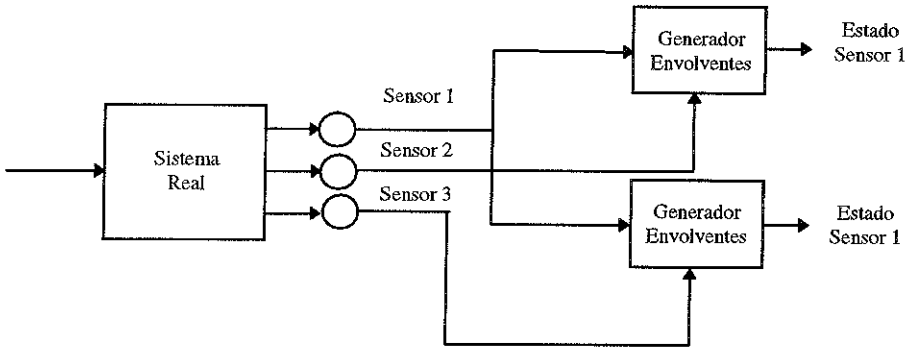


Fig. 11.15 Sistema de Detección y Aislamiento de Fallos de múltiples sensores mediante el algoritmo de generación de envolventes modificado

Este problema se puede resolver utilizando más de un algoritmo de generación de envolventes para cada sensor. En el caso más general, en un sistema con N sensores, se debería utilizar N-1 algoritmos de generación de envolventes para cada sensor, donde cada algoritmo utilizaría todas las medidas disponibles excepto una. El fallo del sensor j-ésimo sería aislado si los N-1 algoritmos aplicados sobre él indicaran el fallo, y los N-1 algoritmos aplicados al resto de sensores del sistema que no utilizan el sensor j-ésimo no lo indicaran. Esta lógica se puede extender a fallos en diversos sensores simultáneos.

11.13 Efecto del ruido en las medidas

Hasta el momento se ha supuesto que el sistema dinámico monitorizado está libre de ruido. Esta suposición es una aproximación correcta en sistemas dinámicos en los cuales el efecto de los errores de modelado es mucho mayor que el efecto del ruido. En muchos sistemas, sin embargo, el ruido en los sensores y en el propio proceso no se puede despreciar. Para incluir el efecto del ruido en los sistemas de detección de fallos basados en el algoritmo de generación de envolventes que hemos presentado en este trabajo existen dos enfoques posibles.

El primero de ellos incluye el efecto del ruido aditivo en el sensor y el proceso dentro del modelo general del sistema dinámico dado por la ecuación

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \qquad (11.84)$$

donde A, B y C son las matrices del sistema, cuyos elementos tienen un valor dentro de un intervalo, simplemente incluyendo a dicho los vectores de ruido. Cada componente de dichos vectores es un intervalo que acota el valor del ruido. Si el ruido se sabe que es de tipo Gaussiano, dichos intervalos quedan fijados a $\pm 3\sigma$. Entonces, el algoritmo de generación de envolventes calcula el intervalo de valores posibles para las medidas obtenidas por el sensor monitorizado incluyendo el efecto aditivo del ruido y de forma que se produzcan falsas alarmas debido al ruido. Es decir, el ruido queda modelo como un offset que es constante durante la ventana temporal de procesado.

Una segunda forma de considerar el efecto del ruido es más apropiada para sistemas en los cuales es necesario un tratamiento estadístico más riguroso. Este enfoque se basa en el hecho de que las

covarianzas del ruido en los sensores calculadas a partir del modelo nominal constituyen una buena aproximación de las covarianzas del ruido en los sensores del sistema monitorizado si las variaciones en los parámetros son relativamente pequeñas. El método de detección de fallos consiste en el cálculo de los intervalos de las medidas de los sensores monitorizados con el algoritmo de generación de envolventes como si el ruido no estuviera presente, y en el cálculo de las covarianzas del ruido en los sensores como si los errores de modelado no estuvieran presentes. El test de la existencia o no de fallo en el sensor se reduce en determinar si el hecho de que una medida proporciona por el mismo esté fuera del intervalo predicho por el algoritmo de generación de envolventes es estadísticamente significativa en presencia del ruido. Por ejemplo, si la mitad de la anchura del intervalo de valores posibles del sensor y la desviación estándar del ruido son iguales, el cruce del límite del intervalo de medidas posibles por la medida del sensor en un único instante de tiempo no es estadísticamente significativo de la existencia de un fallo. Pero si la medida permanece fuera del intervalo durante un largo periodo de tiempo, entonces si que es indicativo de la existencia de un fallo. Existen diversos trabajos de investigación actualmente en marcha (ver [Horak88], [Travé97]) en los cuáles se están desarrollando algoritmos rigurosos de decisión para este problema en particular.

11.14 Método Alternativo de Detección: "Distancia al Espacio de Atractores"

Uno de los problemas que presentan las envolventes cuando son aplicadas a la detección de fallos es que no todos los puntos que están dentro de la envolvente corresponden con situaciones reales posibles, es decir de no fallo, puesto que al utilizar una aproximación de tipo hipercubo de la región de estados posibles se pierde la relación entre los valores que pueden tomar los estados. Para evitar este problema se propone utilizar como indicador de si existe fallo o no primero el que una medida este dentro de la envolvente y a continuación que la distancia al espacio de atractores de la familia de sistemas descrita por el modelo intervalar sea la que corresponde. Para ello utilizaremos los resultados obtenidos en el capítulo 3 donde se formuló matemáticamente el nuevo algoritmo de generación de envolventes.

En primer lugar recordaremos la definición de atractor, así como del conjunto de atractores de un sistema con un modelo con parámetros acotados en intervalos.

La estabilidad del sistema

$$x_{k+1} = A(s)x_k + B(s)u_k \text{ para } s \in S \qquad (11.85)$$

implica que x_k tiende al punto $x(s)$ cuando $t \to \infty$.

Se define un ***punto atractor*** o ***punto de régimen permanente*** del sistema para el parámetro s y $u_k = u$ como el único punto fijo del sistema, por lo tanto:

$$x(s) = A(s)x(s) + B(s)u \qquad (11.86)$$

El ***conjunto atractor*** $E = \{x(s) \mid s \in S\}$ se define como el conjunto de todos los puntos atractores de la familia de parámetros S. Las distancias siguientes nos serán útiles:

$$\delta_k = \max_{x \in X_k} \min_{y \in E} \|x - y\|_\infty \qquad (11.87)$$

$$d_k = \max_{x \in X_k, y \in E} \|x - y\|_\infty \qquad (11.88)$$

Es decir, δ_k representa qué tan cerca está un punto de X_k, conjunto de todos los estados posibles del sistema en el instante k, de su vecino más cercano de E mientras que d_k es la máxima distancia entre puntos de X_k y E.

La envolvente se define como el valor máximo y mínimo para cada uno de los estados del sistema para cada instante de tiempo k. El problema que presentan las envolventes es que contienen combinaciones de estados imposibles de alcanzar por la familia de sistemas que nos describe el modelo con parámetros inciertos.

Como una forma alternativa de determinar si el conjunto de valores de los estados actuales es susceptible de indicar un fallo en el sistema, en lugar de ver si dicho conjunto de estados se encuentra de cada una de las envolventes de cada uno de los estados del sistema, se propone calcular la distancia mínima de cada estado al conjunto de atractores. Si dicha distancia es superior a la máxima distancia teórica a la que deberían encontrarse los estados del sistema para que la envolvente fuera convergente, entonces podemos afirmar que el sistema presenta un fallo.

Matemáticamente lo que se debe calcular es la máxima distancia de cada uno de los estados medidos respecto a los puntos atractores y comparar dicha distancia con la máxima distancia teórica a la que pueden encontrarse los estados del sistema en el instante actual para que la envolvente sea convergente. O sea: existe un punto $y^* \in E$ tal que:

$$d = \min_{y \in E} \|x - y\|_\infty \qquad (11.89)$$

donde x representa los estados del sistema medidos por lo sensores. Por lo tanto:

$$y^* = \arg \min_{y \in E} \|x - y\|_\infty \qquad (11.90)$$

De donde:

$$d = \|x - y^*\| = \max\left\{\left|x_1 - y_1^*\right|, \left|x_2 - y_2^*\right|, \cdots, \left|x_n - y_n^*\right|\right\} \qquad (11.91)$$

bien,

$$d = \|x - y^*\| = \max\left\{x_1 - y_1^*, y_1^* - x_1, x_2 - y_2^*, y_2^* - x_2 \cdots, x_n - y_n^*, y_n^* - x_n\right\} \qquad (11.92)$$

Por lo tanto, los problemas de optimización que se deben resolver son los siguientes:

$$\begin{aligned} &\min x_1 - y_1 \quad \text{s.t.} \quad y \in E \\ &\min y_1 - x_1 \quad \text{s.t.} \quad y \in E \\ &\min x_2 - y_2 \quad \text{s.t.} \quad y \in E \\ &\min y_2 - x_2 \quad \text{s.t.} \quad y \in E \\ &\cdots \end{aligned} \qquad (11.93)$$

quedándonos con el valor máximo de todos ellos.

Si dicho valor máximo supera la distancia máxima a la que pueden estar los estados del sistema en el estado actual para que la envolvente sea convergente, podemos afirmar con toda seguridad que se ha producido un fallo en el sistema. Dicha distancia viene dada por:

$$\delta_k \leq \xi_l d_{k-1} \qquad (11.94)$$

con $\xi_l = \kappa(s)\rho(s)^1$ siendo κ el número de condición de la familia de matrices A y ρ su radio espectral.

Veamos un ejemplo concreto para ver como podemos aplicar este variante del nuevo algoritmo de detección de fallos. Supongamos que el sistema es un sistema de segundo orden que posee el siguiente modelo en el espacio de estado:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

con:

$$a_1 \in [1.3938, 1.4338]$$

$$a_2 \in [0.5865, 0.6265]$$

El conjunto de atractores de este sistema cuando la entrada u(k) = 1 se puede determinar de la siguiente manera:

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} a_1 & -a_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

de donde operando se llega a:

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} \dfrac{-1}{a_1 - 1 - a_2} \\ \dfrac{-1}{a_1 - 1 - a_2} \end{bmatrix}$$

cuya representación gráfica es la que se muestra en la figura:

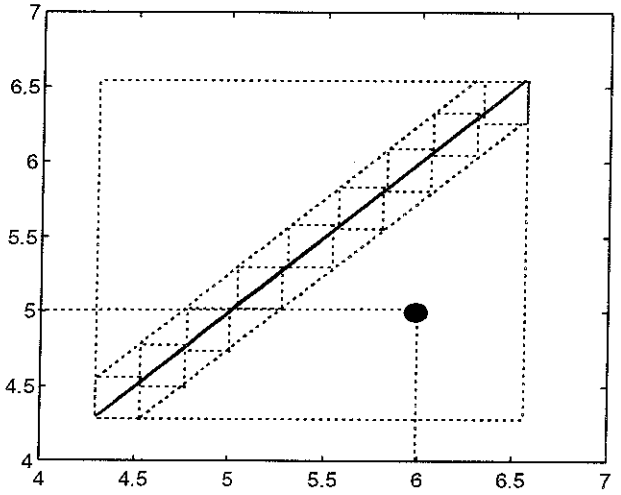


Fig. 11.16 Espacio de atractores y obtenida por el sensor

En régimen permanente la envolvente vale:

$$x_1 \in [4.2974, 6.5488]$$

$$x_2 \in [4.2974, 6.5488]$$

Para ver como funciona el cálculo de la distancia al punto de atractores, supongamos que la medida que nos proporciona el sensor para k = 5 es: $x_1 = 6$ y $x_2 = 5$.

Para este mismo instante de tiempo las correspondientes envolventes de dichos estados toman los siguientes valores:

$$x_1 \in [5.1865, 6.1219]$$

$$x_2 \in [4.6713, 5.1688]$$

por lo tanto, la situación medida por los sensores es de no fallo cuando en realidad es de fallo.

Para determinar la mínima distancia de la medida tomada con los sensores respecto al espacio de atractores se deben resolver los siguientes problemas de optimización:

$\min \quad d_1 = x_1 - y_1$ $\text{s.t.} \quad y_1 + \frac{1}{a_1 - 1 - a_2} = 0$ $y_2 + \frac{1}{a_1 - 1 - a_2} = 0$ $a_1 - a_1^+ \le 0$ $a_1^- - a_1 \le 0$ $a_2 - a_2^+ \le 0$ $a_2 - a_2^+ \le 0$	$\min \quad d_2 = -x_1 + y_1$ $\text{s.t.} \quad y_1 + \frac{1}{a_1 - 1 - a_2} = 0$ $y_2 + \frac{1}{a_1 - 1 - a_2} = 0$ $a_1 - a_1^+ \le 0$ $a_1^- - a_1 \le 0$ $a_2 - a_2^+ \le 0$ $a_2 - a_2^+ \le 0$
$\min \quad d_3 = x_2 - y_2$ $\text{s.t.} \quad y_1 + \frac{1}{a_1 - 1 - a_2} = 0$ $y_2 + \frac{1}{a_1 - 1 - a_2} = 0$ $a_1 - a_1^+ \le 0$ $a_1^- - a_1 \le 0$ $a_2 - a_2^+ \le 0$ $a_2 - a_2^+ \le 0$	$\min \quad d_4 = -x_2 + y_2$ $\text{s.t.} \quad y_1 + \frac{1}{a_1 - 1 - a_2} = 0$ $y_2 + \frac{1}{a_1 - 1 - a_2} = 0$ $a_1 - a_1^+ \le 0$ $a_1^- - a_1 \le 0$ $a_2 - a_2^+ \le 0$ $a_2 - a_2^+ \le 0$

Por otro lado, utilizando las expresiones que nos proporcionan la máxima distancia a la que podemos estar de cualquier punto del espacio de atractores en cada instante de tiempo para que la envolvente sea convergente:

$$\delta_k \le \xi_l d_{k-l}$$

donde:

$$\xi_l = \kappa(s)\rho(s)^l$$

$\kappa(s)$ es el número de condición de la familia de matrices A(s) y $\rho(s)$ es su radio espectral.
Así, por ejemplo para k=5 con l=20 tendremos:

$$\delta_5 \le \xi_{20} d_0$$
$$\xi_5 = \kappa(s)\rho(s)^{20}$$

En este ejemplo:

$$d_o = 6.5488$$
$$\xi_5 = 0.0611$$

por lo tanto, la distancia mínima al espacio de atractores debería ser:

$$\delta \le 0.3999$$

Por otro lado, se ha determinado la ventana de longitud mínima estable y la ventana mínima casi estacionaria obteniéndose los siguientes resultados:

$$L_\infty = 7$$
$$L_{0.05} = 20$$

Una vez determinada la distancia teórica a la que deberíamos encontrarnos en un determinado estado de tiempo vamos a calcular utilizando la TOOLBOX de Optimización de MATLAB la distancia mínima a los puntos atractores a la que efectivamente se encuentra el sistema utilizando como referencia la medida de los estados del sistema obtenidos mediante los sensores. Si suponemos que los sensores nos han proporcionado la siguiente medida: $x_{1m} = 6$ y $x_{2m} = 5$, entonces la solución a los problemas de optimización anteriores que nos proporciona MATLAB, para el caso en que mantegamos x_2 constante y busquemos el punto x_1 del espacio de atractores que se encuentra a mínima distancia del estado real medido, es la siguiente:

$x_1=5$
$x_2=5$
$a_1=1.4000$
$a_2=0.6000$

A cotinuación, mantendremos la x_1 constante y buscaremos la x_2. La solución que nos proporciona MATLAB es la siguiente:

$x_1=6$
$x_2=6$
$a_1=1.4232$
$a_2=0.5898$

Por lo tanto la mínima distancia al espacio de atractores es 1, para los dos casos. Con lo cual podemos afirmar que la medida obtenida es indicativa de un fallo puesto que la mínima distancia para este caso debería ser

$$\delta \leq 0.3999$$

11.15 Conclusiones

En este capítulo se presenta la generación de envolventes como el mecanismo para la generación de umbrales adaptativos necesario para construir un sistema de detección y diagnóstico de fallos robusto a partir del modelo del sistema con incertidumbre intervalar en los parámetros del mismo. La integración de la técnica de generación de envolventes dentro del sistema de detección y diagnóstico de fallos se ha propuesto llevarla a cabo de la siguiente manera:

- utilizando la generación de residuos utilizando la técnica de las ecuaciones de paridad, aunque también como de ha demostrado más adelante en este capítulo sería también posible llevarla a cabo utilizando la técnica de generación de residuos basada en observadores.
- una vez generados los residuos su valor debe ser evaluado mediante un umbral que para nuestro caso será un umbral dinámico obtenido mediante las envolventes.
- finalizada la evaluación de los residuos estos se pueden combinar adecuadamente para obtener una matriz de incidencia con las propiedades de aislabilidad de los fallos deseada.
- así mismo, utilizando la técnica DMP se obtendrá la sensibilidad de cada uno de los residuos respecto a cada uno de los posibles fallos que se desean detectar, de forma que combinando el vector de residuos evaluados, al que DMP denomina, vector de satisfacción con la matriz de sensibilidad, se obtendrá el vector de fallos, a partir del cual, se podrá diagnosticar la existencia de un fallo.

La técnica de generación de umbrales adaptativos a partir de las envolventes por lo tanto es fácilmente integrable dentro de un sistema de detección y diagnóstico de fallos, proporcionando además robustez a dicho sistema frente a la incertidumbre existente sobre el valor de los parámetros del modelo, y tal como hemos visto en el primer capítulo de esta tesis, minimizando el número de falsas alarmas debidas a errores de modelado, y maximizando el número de fallos detectados a la vez que es posible detectar los mínimos fallos posibles teniendo en cuenta la incertidumbre existente en el modelo del sistema.

Capítulo 12

Capítulo 12:
CONCLUSIONES Y TRABAJO FUTURO

12.1 Conclusiones Generales y Aportaciones de la Tesis

En esta tesis se ha presentado un nuevo algoritmo para la generación de envolventes a partir del modelo del sistema con incertidumbre estructurada. La generación de envolventes es una de las posibles técnicas para la generación de umbrales adaptativos dentro de los métodos de detección robustos, y en concreto, dentro de los métodos robustos pasivos.

Después de realizar un repaso al estado del arte tanto sobre los métodos de detección robustos como sobre los métodos de generación de envolventes podemos afirmar que el método de generación de envolventes propuesto en esta tesis es novedoso y además obtiene unas envolventes exactas, o bien, con la aproximación deseada por el usuario a diferencia del resto de métodos existentes hasta el momento.

El nuevo método de generación de envolventes soluciona los problemas que padecen la mayoría de métodos de generación de envolventes existentes y que son:

- el wrapping
- el problema de las multiincidencias
- el problema de los óptimos locales
- el problema de la propagación de la incertidumbre

todo ello gracias a que el nuevo algoritmo tiene en cuenta que para que el método utilizado para la generación de envolventes no aumente la incertidumbre ya existente en el sistema debe de partir siempre del la incertidumbre existente en el estado inicial del mismo donde se supone bien conocida, o bien, si no se parte del instante inicial partir de L muestras hacia atrás donde L es la longitud de una ventana temporal deslizante y su valor tal como se ha demostrado en esta tesis es del orden del tiempo de establecimiento del sistema. Es decir, la ventana temporal debe capturar el régimen transitorio del sistema. Al partir, por lo tanto, de L pasos hacia atrás el problema del wrapping, es decir, la inclusión de estados espúreos durante la generación de las envolventes debido a la aproximación en forma de región hiperrectangular del espacio de estados posibles también desaparecerá tal como se ha demostrado tanto de forma analítica como mediante simulaciones.

El problema de las multiincidencias y de los óptimos locales se evita utilizando técnicas de optimización globales que garantizan de forma rigurosa que se obtendrá el óptimo global con la precisión deseada por el usuario.

Al evitar por lo tanto el problema del wrapping, el problema de las multiincidencias y el problema de los óptimos locales también se evitará del problema de la propagación de la incertidumbre, evitándose la producción de envolventes inestables. La incertidumbre añadida por el nuevo algoritmo está controlada y el usuario puede decidir que cantidad de incertidumbre admite en aras a mejorar el tiempo de cálculo.

La tesis ha supuesto en todo momento que el modelo del sistema que se estaba utilizando era el modelo lineal y que los parámetros entre instantes de tiempo se mantenían constantes aunque su valor podía no ser conocida sabiéndose sólo que estaba acotado dentro de un intervalo.

12.2 Conclusiones sobre el Nuevo Algoritmo de Generación de Envolventes

Después de realizar un repaso a todos los algoritmos existentes de generación de envolventes en la bibliografía se llega a la conclusión que ninguno de ellos obtiene unas envolventes correctas al cien por

cien, todos los métodos tienen algún punto débil. El método que presentamos en esta tesis tiene la ventaja de que puede llegar a producir las envolventes correctas y exactas utilizando una ventana temporal suficientemente larga. Básicamente los dos puntos clave del nuevo algoritmo y que permiten afirmar que el nuevo algoritmo puede generar unas envolventes correctas y exactas son:

- el no propagar la incertidumbre de un paso al siguiente sino el utilizar una ventana temporal que permite si es suficientemente larga que el efecto de la incertidumbre de los estados al inicio de la misma haya desaparecido al final de la misma, de la misma manera que desaparece el efecto de las condiciones iniciales en el régimen permanente de un sistema.
- el utilizar técnicas de optimización global que permiten explorar todo el espacio de valores de parámetros y estados asegurando que el valor de la envolvente calculada es el correcto.

Los aspectos que "a priori" pueden mejorarse son:

- el tiempo de cálculo del algoritmo puesto que al utilizar la ventana temporal y para sistemas de orden superior la expresión de la función objetivo puede alcanzar un tamaño difícil de manejar por los solvers de optimización
- el tamaño de las expresiones a manejar y su complejidad puede llegar a hacer este nuevo algoritmo intratable para sistemas de orden superior, aunque éste no es un problema específico del algoritmo que presentamos en esta tesis si no en general de todos los métodos de generación de envolventes, pero al trabajar con una ventana temporal este problema se agudiza puesto que las expresiones tienen una longitud superior al tener que considerar los valores de los estados desde el inicio de la ventana temporal.

12.3 Conclusiones sobre la Formulación Matemática del Nuevo Algoritmo

Uno de los puntos fuertes del nuevo algoritmo de generación de envolventes es que se ha conseguido una formulación matemática del mismo que ha permitido demostrar de forma rigurosa su correcto funcionamiento a la vez que se ha conseguido determinar analíticamente uno de los parámetros claves del mismo: la longitud de la ventana temporal.

Para realizar dicha demostración lo que se han utilizado herramientas básicas de álgebra. Básicamente, la formulación matemática del nuevo algoritmo consiste en intentar demostrar que longitud de ventana se debe utilizar para que:

- en primer lugar, la incertidumbre del sistema debido al algoritmo de generación de envolventes no se propague de forma descontrolada generando unas envolventes inestables. A esta longitud de ventana la hemos denominada mínima longitud de ventana estable.
- en segundo lugar, la incertidumbre del sistema no aumente más de un cierto porcentaje respecto al caso exacto, es decir, el caso en el que se utilizara una ventana temporal infinita. A esta longitud de ventana la hemos denominado longitud de ventana casi estable para un determinado un incremento de incertidumbre.

Las longitudes de ventana necesarias obtenidas a partir de la formulación matemática del algoritmo de generación de envolventes se han utilizado sobre varios ejemplos de aplicación comprobándose que efectivamente dan los resultados correctos.

Intuitivamente, la longitud de la ventana correcta para producir unos resultados parecidos a los obtenidos con el método exacto parece que debería ser del orden del tiempo de establecimiento. Recordemos que este tiempo representa el tiempo necesario que necesita el sistema para que alcanzar desde el último cambio de consigna el 2% ó el 5% del valor final. Efectivamente, en este caso la intuición se confirma al generalizar la formulación del algoritmo a tiempo continuo. Obteniéndose, que efectivamente la longitud de la ventana para obtener un resultado equivalente al obtenido con la ventana de longitud infinita es del orden del tiempo de establecimiento, pero además, se comprueba que la longitud de ventana mínima estable es del orden de la constante de tiempo del sistema, un resultado nuevo que no se ha había intuido, pero que se puede comprobar mediante simulaciones.

12.4 Conclusiones sobre la Resolución del Problema de Optimización asociado

Tal como hemos dicho al principio de este capítulo uno de los puntos claves del nuevo algoritmo es la formulación del problema de la generación de las envolventes como un problema de optimización, de forma que el problema de generación de envolventes se trasnforma en un problema matemáticamente bien formulado, y la solución del mismo permitirá obtener las envolventes correctas. El problema es que, si bien la formulación del problema de optimización es simple pero a la vez sólida y rigurosa, incluyendo todos los comportamientos posibles del sistema y representándolos de una forma compacta, la resolución del mismo no es tan simple. De entrada el campo de la Optimización aún es un campo en plena investigación, la mayoría de los métodos existentes sólo pueden garantizar que la solución alcanzada es un óptimo local y no un óptimo global. Ello se debe a dichos algoritmos basan la búsqueda de la solución del problema de optimización en la búsqueda en el valor de la pendiente de la función objetivo obtenido a partir de la derivada partiendo de un punto semilla. Por lo tanto, si se escoge incorrectamente el punto semilla tomándolo cercano a una solución local, el algoritmo de optimización obtendrá como solución dicha solución local. Sólo para el caso en que el problema de optimización reúna las siguientes propiedades:

- función objetivo convexa
- región factible determinada por las restricciones también convexa

es decir, sólo para el caso de problemas de optimización convexos, para los cuales sólo existe una solución, y por lo tanto, la solución local coincide con la global, los métodos de optimización clásicos pueden garantizar que el óptimo alcanzado es el global. Sin embargo, el problema de optimización que aparece al aplicar el nuevo algoritmo de generación de envolventes no es un problema convexo, puesto que si bien la región factible es una región convexa, se trata de un región hipercúbica, la función objetivo no es general convexa puesto que se trata de una función lineal de tipo polinomio de varias variables, que en general contendrá coeficientes positivos y negativos.

Por lo tanto, los métodos de optimización clásicos basados en búsqueda local en el espacio de soluciones no nos son útiles para la resolución del problema de generación de envolventes mediante el nuevo algoritmo, puesto que podremos asegurar que los óptimos alcanzados se correspondan con óptimos globales, con lo cual se obtendrían envolventes incorrectas. Al alcanzar un óptimo local en lugar de uno de global se estarían obteniendo unas envolventes subdimensionadas, perdiéndose muchos estados posibles. Se ha debido, por lo tanto, recurrir a métodos que nos puedan garantizar de forma rigurosa y con la exactitud deseada el óptimo global, debiéndonos pues introducir dentro de una de las disciplinas en la que en los últimos años la investigación ha sido muy intensa, me refiero al campo de la Optimización Global. Se trata de la disciplina que estudia métodos y algoritmos para determinar óptimos globales de problemas de optimización. Evidentemente las técnicas utilizadas son diferentes de las utilizadas por los métodos de Optimización Clásica (Local). Existen dos grandes familias de técnicas: las técnicas deterministas y las técnicas estocásticas. En principio, en nuestro caso nos hemos centrado en las técnicas deterministas puesto que son las únicas que pueden garantizar de forma rigurosa que alcanzarán el óptimo global, aún que el número de pasos necesario no se puede predecir, puesto que depende del tipo de problema. Dentro de las técnicas deterministas existen también multitud de técnicas, aunque una de las más utilizadas en la técnica de "branch ad bound", que consiste en una búsqueda exhaustiva dentro de un árbol binario, del cual se van descartando casos a partir de la realización de pruebas sobre cada uno de los nodos para comprobar su factibilidad o infactibilidad como solución del problema.

El problema de optimización que aparece al utilizar el nuevo algoritmo de generación de envolventes ha debido, por lo tanto, ser resuelto por alguna técnica de optimización determinista y basada en branch and bound. Debido a la naturaleza del problema de optimización al que nos enfrentamos: función objetivo no lineal y de tipo polinómico, este tipo de problema de optimización se conoce también como problema de optimización geométrico. Los problemas de optimización geométricos aparecen en multitud de problemas de ingeniería. En concreto, cuando los coeficientes de la función polinómica pueden tomar valores positivos y negativos, el problema de optimización geométrico se denomina problema de optimización signomial. Su solución global es posible mediante el algoritmo de Falk, basado en una

búsqueda de tipo branch and bound sobre problemas convexos obtenidos mediante relajaciones sucesivas del problema original. Cada uno de los problemas convexos obtenidos deben de ser resueltos mediante un algoritmo de optimización clásico. Se trata de un método que si bien garantiza que se alcanza el óptimo global, su convergencia es muy lenta y además la complejidad de cálculo es alta debido a que cada uno de los problemas convexos deben de ser resueltos mediante la aplicación de un algoritmo de optimización clásico.

El problema de optimización a resolver puede calificarse también como un problema no separable, puesto que, si bien la función objetivo es no lineal, ésta es de tipo polinómico, es decir, formada por sumas de monomios. Cada uno de estos monomios es su vez una función no lineal. Por lo tanto, la función objetivo está compuesta por la suma de funciones no lineales. Los problemas separables pueden ser resueltos de forma global utilizando una aproximación lineal a trozos de cada una de las funciones no lineales, formulándose un problema aproximado y lineal a trozos. Este tipo de problema se puede formular como un problema de programación entera mixta, que para resolverlo, se ha utilizado, de nuevo, el algoritmo de branch and bound sobre un conjunto de nodos, dada uno de ellos, representado un posible problema de optimización lineal, que se podrá, por lo tanto, resolver como un problema de programación lineal. Este método es bastante adecuado para resolver el problema de generación de envolventes, el problema estriba en preparar el problema a resolver linealizándolo a trozos y traduciéndolo a un formato adecuado para que un solver de programación entera mixta pueda resolverlo. En cuanto al tiempo de resolución del problema, una vez preparado es aceptable para problemas de tamaño normal.

Un método alternativo a los métodos basados en la naturaleza del problema son los métodos que utilizan el álgebra de intervalos como método de acotación del rango de la función junto con el algoritmo de búsqueda branch and bound. Estos métodos de los cuales uno de los algoritmos más conocidos es el algoritmo de Hansen, permiten obtener el óptimo global de la función objetivo del problema de generación de envolventes de una forma bastante rápida y eficiente, siendo éste el método que mejores resultados nos ha proporcionado de los que se han probado. Así mismo parece que en el futuro la mejora de este tipo de algoritmos continuará intentado aumentar su velocidad de convergencia y reduciendo el tiempo de cálculo, introduciendo mejoras en el álgebra de intervalos que tengan en cuenta la interactividad entre las variables que hacen dicha álgebra tenga tendencia a sobreestimar el rango de la función, utilizando procesadores en paralelo, etc.

Finalmente, se ha propuesto plantear el nuevo algoritmo de generación de envolventes como un problema de satisfacción de restricciones intervalares. Esta propuesta es muy atractiva puesto que ofrece una formulación del problema a resolver mucho más compacta, puesto que el problema de generación de envolventes de un sistema, se traduce en un único problema de satisfacción de restricciones intervalares cuya solución a cada instante nos proporciona directamente el valor de las envolventes para cada uno de los estados del sistema. En cambio, con las técnicas de optimización global anteriores se debería resolver dos problemas de optimización global para cada uno de los estados del sistema, uno para la envolvente superior y otros para la envolvente inferior. Para resolver el problema de satisfacción de restricciones intervalares se utilizará el algoritmo de propagación de tolerancia global de Hyvönen. Este algoritmo se basa en la obtención de una agenda de funciones solución globales, que pueden ser resueltas dependiendo de si son cíclicas o acíclicas, es decir, de si contienen o no multiincidencias con el algoritmo de propagación de tolerancia local o con un algoritmo de optimización global basado en branch and bound y álgebra de intervalos. Los tiempos de cálculo obtenidos con este método superan en general a los obtenidos con utilizando directamente los algoritmos de optimización global basados en branch and bound y álgebra de intervalos.

Después del estudio exhaustivo de técnicas de optimización llevado a cabo a lo largo de esta tesis, si tuviera que apostar hoy en día por un algoritmo de optimización global de entre los estudiados escogería el algoritmo de optimización global basado en branch and bound y álgebra de intervalos debido a su mayor generalidad, prestaciones y velocidad de cálculo obtenido, además su facilidad de utilización. Un problema de optimización global tratado con este tipo de optimizador no precisa un tratamiento previo tan complicado como el necesario en un algoritmo como el de programación entera mixta. O al menos, dicho tratamiento queda incluido dentro del optimizador.

Un aspecto que todos estos métodos de resolución del problema de optimización asociado al algoritmo de generación de envolventes tienen en cuenta es la precisión con la que se desea obtener la solución. De ante mano el usuario de dichos algoritmos puede fijar con que precisión desea obtener el óptimo (máximo o mínimo) de la función objetivo. De esta forma se puede reducir la precisión del resultado para así reducir el tiempo de cálculo, llegándose a un compromiso entre ambos. Lo que si es evidente es que a mayor precisión mayor tiempo de cálculo. Recordemos que el algoritmo de generación de envolventes debe de poder funcionar en tiempo real para poder ser útil como mecanismo de detección de fallos, objetivo último del mismo. Es por ello, a que podemos preferir no obtener tanta precisión en el resultado para reducir el tiempo de cálculo, pero en todo, momento sabremos el resultado exacto dentro de que franja se encuentra. Es decir, resultado final obtenido por el algoritmo de optimización nos proporcionará para cada valor de la envolvente (inferior o superior) un intervalo de anchura dos veces la precisión fijada, que deberá de ser tenido en cuenta por el algoritmo de detección de fallos basado en la envolvente. Esto significa, que el algoritmo de detección basado en la envolvente no deberá comprobar sólo si la medida obtenida del estado del sistema está dentro de la envolvente sino que también si está o no dentro del intervalo asociado con la envolvente superior e inferior.

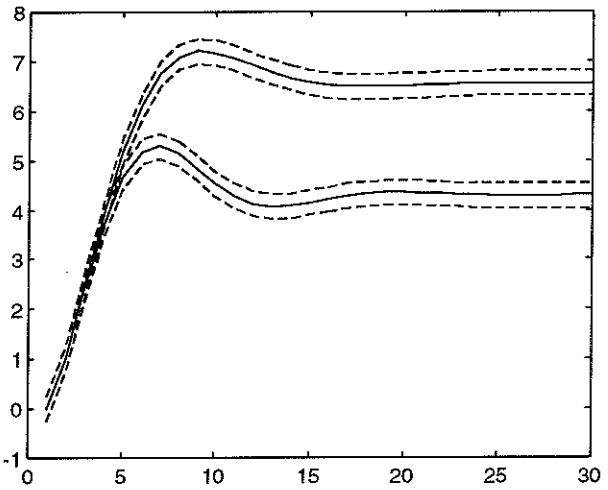


Fig. 12.1 Envolventes con la franja de incertidumbre debida a la precisión en la optimización

12.5 Conclusiones sobre la Integración con Métodos de Diagnóstico

Una vez las envolventes del sistema han conseguido ser generadas adecuadamente, estas envolventes pueden ser utilizadas tal como se ha visto en los primeros capítulos de esta tesis como umbrales adaptativos a utilizar en el momento de la detección. Según se ha visto cuando se utiliza el modelo incierto de un sistema para utilizarlo para detectar fallos en el mismo, para que el método de detección sea robusto frente a esta incertidumbre el método de detección debe de tenerla en cuenta en el momento de realizar dicha detección. La incertidumbre puede ser tenida en cuenta en el momento de la generación del residuo, en este caso hablamos de robustez activa, o bien, en el momento de la comparación del residuo con el umbral de detección, en este caso hablamos de robustez pasiva. Las envolventes al utilizarlas como umbrales de detección constituyen por lo tanto un método de detección robusto pasivo. Los umbrales de detección a utilizar para el caso de utilizar modelos con incertidumbre dependen del punto de funcionamiento del sistema, así como de si el sistema en régimen permanente o en régimen transitorio, por lo tanto, no pueden ser umbrales fijos sino que deben ir cambio con el punto de funcionamiento y con el régimen del sistema. Dichos umbrales con estas características son los que se consiguen con las envolventes.

Así mismo, al utilizar un método de diagnóstico junto con el método de detección de fallos basado en las envolventes dicho método de diagnóstico utiliza también umbrales para considerar a los residuos como significativos y para medir los significativos que son. De entre estos métodos, uno de los más utilizados

es el algoritmo DMP. Dicho algoritmo contempla sólo la fase de análisis de los residuos, pero no contempla ni su generación ni su evaluación. En este trabajo proponemos un algoritmo DMP mejorado que incluya los métodos de generación de residuos estructurados basados en ecuaciones de paridad, como método de evaluación de los mismos las envolventes y como método de diagnóstico el algoritmo DMP pero utilizando como tolerancias a utilizar para evaluar el grado de significación de los residuos la tolerancia obtenida a partir de las envolventes.

La estructura del sistema de detección y diagnóstico propuesto se muestra a continuación en la Fig. 12.2. En dicho diagrama pueden apreciarse tres bloques importantes: la generación del residuo, la evaluación del mismo mediante la generación de las envolventes y finalmente el diagnóstico del fallo mediante el análisis de una batería de residuos estructurados.

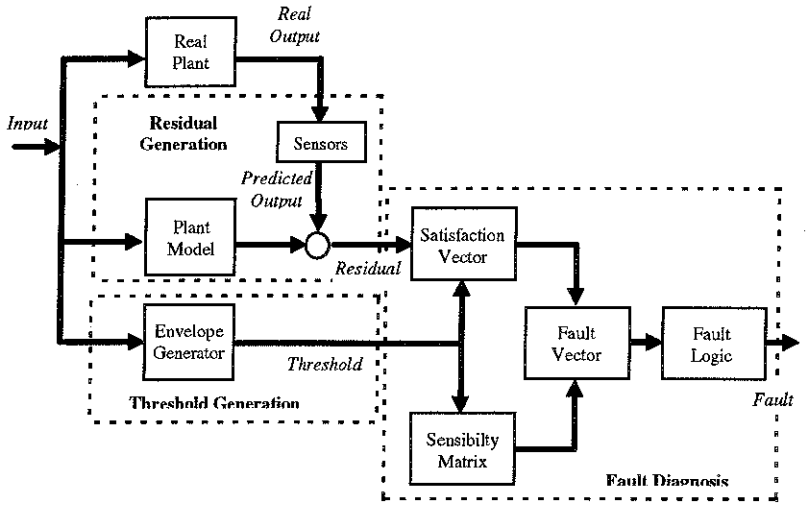


Fig 12.2 Diagrama de bloques del sistema de detección y diagnóstico de fallos

12.6 Trabajo Futuro

Después de todos los temas abordados a lo largo de esta tesis han quedado muchos puntos abiertos que podrán ser abordados en futuros trabajos de investigación o tesis. De hecho en determinados aspectos, como por ejemplo, en el tema de integración del algoritmo de generación de las envolventes con algoritmo de diagnóstico la tesis sólo ha apunto una posible vía de integración debiéndose de explorar con mayor profundidad y rigor.

Dentro de los puntos que a mi entender merecen una mayor profundización son los siguientes:

- obtención de los modelos con parámetros inciertos
- integración del algoritmo de generación de envolventes con algoritmos de detección
- estudio del problema del ruido en los sensores
- corrección de la predicción efectuada por la envolvente de los estados del sistema con medidas de sensores
- estudio del algoritmo de detección basado en la distancia al espacio de atractores en regimen permanente
- el problema de trabajar con la incertidumbre a diferentes niveles: modelos, transformaciones, etc.
- mejora del tiempo de cálculo del algoritmo de generación de envolventes mediante diversas técnicas.
- aplicaciones reales para validar y contrastar los resultados teóricos
- realización de benchmark con otras técnicas de generación de umbrales adaptativos

A continuación vamos a desgranar un poco cada uno de estos puntos:

12.6.1 Obtención de los Modelos con Parámetros Inciertos

A lo largo de toda esta tesis se ha partido de la suposición de que se disponía de un modelo con parámetros inciertos. Pero las pregunta son: ¿cómo se ha obtenido dicho modelo? ¿qué técnicas existen para obtener modelos con parámetros inciertos?

El tema del modelado de sistemas con parámetros inciertos es aún hoy un tema abierto de investigación. Un primer enfoque posible a dar al tema es intentar generalizar las técnicas de identificación de sistemas tradicionales basadas en criterios estadísticos, básicamente en técnicas relacionadas con los mínimos cuadrados, para a partir de los resultados obtenidos por dichas técnicas obtener el modelo del sistema con parámetros inciertos. Puesto que dichas técnicas nos proporcionan para cada uno de los parámetros del sistema su valor medio y su desviación estándar, podríamos suponer que el modelo con parámetros inciertos se podría obtener tomando como acotación de la incertidumbre la desviación estándar proporcionada por los métodos de identificación estadísticos. Las pruebas realizadas durante esta tesis sobre esta forma de obtener los modelos con parámetros inciertos han proporcionado resultados decepcionantes, puesto que los comportamientos del modelo aproximado obtenido engloban muchos más comportamientos que los realmente observados.

Otra forma de atacar el tema de los modelos de sistemas con parámetros inciertos es recurrir a técnicas de identificación deterministas.

12.6.2 Integración de la Generación de Envolventes con Algoritmos de Diagnóstico

Al final de esta tesis se ha realizado una propuesta de posible integración del la generación de umbrales adaptativos mediante envolventes con algoritmos de diagnóstico para así de esta forma desarrollar un sistema de detección y diagnóstico completo. Creemos que el estudio de tales tipos de sistema merece una mayor profundidad para estudiar aspectos como: mínimas fallos detectables, índice de fallos perdidos, etc. Así como comprobar mediante aplicaciones prácticas la bondad de este tipo de sistemas de detección y diagnóstico de fallos comparándolos con otras estrategias ya sean basadas en modelos ya sean basadas en el conocimiento.

12.6.3 Estudio del Problema del Ruido, Imprecisión y Fallos en los Sensores

Un problema que siempre está presente cuando tratamos de temas de detección de fallos es el problema del ruido e imprecisión en los sensores. Sin embargo, sin bien se trata de un tema omnipresente nunca se le dedica la atención que creo que se merece estudiando como evitar o tener en cuenta en la medida de lo posible su efecto, de forma que el esfuerzo que se realiza al tener en cuenta la incertidumbre en el modelo del sistema utilizando por ejemplo un umbral adaptativo generado obtenido a partir de las envolventes del sistema no quede enmascarado por el efecto del ruido e imprecisión en los sensores. Así mismo resulta interesante si el tema de detección de fallos en sensores debe ser tratado de forma hardware mediante técnicas de redundancia física de sensores, o bien, es más adecuado su tratamiento mediante técnicas de redundancia software.

12.6.4 Corrección de la Predicción de las Envolventes mediante Lecturas de Sensores

Durante la mayor parte de la tesis se ha considerado que las envolventes generadas a partir del modelo del sistema con parámetros inciertos no eran corregidas en ningún momento por la lectura de los sensores. En los capítulos finales se ha sugerido una posible forma de corrección de las predicciones de los valores de los estados del sistema mediante lecturas de sensores disponibles en el sistema utilizando la estructura de un observador. Creemos que esa es la vía correcta puesto que ofrece mayor grado de libertad al diseñador del sistema de detección de fallos que el de efectuar un reset completo de los estados del sistema en base a las medidas obtenidas por los sensores. Creo que este tema debería ser objeto de estudio y sobre todo de experimentación en aplicaciones reales.

12.6.5 Estudio del Algoritmo de Detección basado en la Distancia al Espacio de Atractores

Al final del capítulo anterior se ha propuesto un nuevo algoritmo de detección basado en el cálculo de la distancia de los estados reales del sistema al espacio de atractores del sistema en regimen permanente y su comparación con la distancia teórica a lo que dichos estados deberían encontrarse utilizando los resultados obtenidos en el capítulo de dedicado a la formalización matemática del algoritmo de generación de envolventes. Este nuevo algoritmo de detección debe ser considerado con mayor atención que la se ha dedicado en esta tesis, puesto que resulta un enfoque muy atractivo y puesto que además consigue completar el enfoque de la detección realizada sólo mediante las envolventes. Si se utilizan sólo las envolventes como método de detección de fallos medidas de los estados del sistema que caen dentro de las mismas no asegura la no existencia de fallo en el sistema, debido a que la envolvente realiza una aproximación hiperrectangular del espacio real de estados posibles. En cambio, la detección basado en el cálculo de la distancia al espacio de atractores es un indicador más preciso de si el sistema se encuentra o no en fallo que la simple comprobación de si los estados del sistema se encuentran dentro del hiperrectángulo definido por las envolventes.

12.6.6 Mejora del Tiempo de Cálculo del Nuevo Algoritmo de Generación de Envolventes

El nuevo algoritmo que propone esta tesis creemos que proporciona unos resultados lo suficientemente buenos como para dedicar un mayor esfuerzo en la mejora del tiempo de cálculo. Para conseguir dicha mejora se proponen las siguientes estrategias a estudiar:

- *Efecto del tiempo de muestreo* sobre la longitud de la ventana temporal. Tal como hemos visto a lo largo de esta tesis la longitud de la ventana temporal es una pieza clave para el buen funcionamiento del algoritmo de generación de las envolventes. Pero existe un compromiso entre la longitud de la ventana, el tiempo de cálculo y la incertidumbre añadida a las envolventes generadas. Por lo tanto, la reducción de dicha longitud sin afectar la incertidumbre incorporada a las envolventes es un objetivo deseable. Una de las forma de reducir dicha longitud es la de utilizar un tiempo de muestreo mayor. Recordemos que en control digital existe una regla que nos indica la velocidad a la que debe realizarse el muestreo para que el modelo obtenido sea el adecuado para realizar el control. Dicho tiempo se obtiene como una fracción del tiempo de subida del sistema, en concreto, la franja de valores posibles es la siguiente:

$$\frac{t_r}{15} \leq T_s \leq \frac{t_r}{4}$$

  El valor usual que se toma es $T_s = \frac{t_r}{10}$. Es decir, se toman unas diez muestran durante el tiempo de subida del sistema. Por otro lado, hemos visto que la longitud L de la ventana es del orden del tiempo de establecimiento t_s. Puesto que el tiempo de establecimiento es del orden 2,5 veces el tiempo de subida t_r, ello implica que durante el tiempo de establecimiento se toman unas 25 muestras, con lo cual la longitud de la ventana es del orden de 25. Si en cambio en lugar de tomar como tiempo de muestreo la décima parte del tiempo de establecimiento tomáramos la cota inferior de los valores posibles, o sea, un cuarto ello implicaría que tendríamos cuatro muestras durante el tiempo de subida del sistema, o bien, aproximadamente, 10 muestras durante el tiempo de establecimiento. Con lo cual la longitud de la ventana se reduciría hasta 10 muestras. Los tiempo de cálculo obtenidos con dicha longitud son más que aceptables, del orden 33 ms para un sistema de segundo orden utilizando el algoritmo de optimización global basado en branch and bound y álgebra de intervalos.

- *Efecto de la aproximación hipercúbica* del espacio de estados sobre la longitud de la ventana temporal. Otra posible forma de reducir la longitud de la ventana temporal es intentar utilizar una mejor aproximación de la región de estados posibles. Recordemos que el efecto de la ventana temporal es eliminar el efecto de los estados espúreos contenidos dentro de la caja hipercúbica en la que encerramos la región de estados posibles. Si en lugar de encerrar la región de estados posibles en una caja hipercúbica utilizáramos un poliedro convexo que se ajustara mejor a la región real de estados posibles, el número de estados espúreos a tener en cuenta por el algoritmo de generación de envolventes sería menor, con lo cual su eliminación también sería más fácil, requiriéndose una ventana temporal menor. Además recordemos que el efecto wrapping era provocado por encerrar la

región de estados posibles mediante una caja hipercúbica. Utilizando otro tipo de caja se podría mitigar este efecto.

- *Paralelización del algoritmo de generación de las envolventes.* Puesto que se deben de resolver dos problemas de optimización diferente para cada estado del sistema, uno para la envolvente superior y otro para la envolvente superior, ello implica que para un sistema de orden n se deben resolver para cada instante de tiempo 2n problemas de optimización simultáneos, que podrían ser llevados a cabo por 2n procesados en paralelo. Por lo que acabamos de comentar, el algoritmo de generación de envolventes se trata de un sistema fácilmente paralelizable, puesto que además estos 2n problemas de optimización son independientes entre sí, con lo cual no existe penalización en el intercambio de información entre los procesadores puesto que cada problema se puede resolver independientemente de los demás.

- *Paralelización del algoritmo de optimización global.* Si utilizamos tal como hemos dicho el algoritmo de optimización global basado en el algoritmo de branch and bound y álgebra de intervalos podemos pensar también paralelizar dicho algoritmo. De hecho existen diversos estudios y tesis sobre la posible paralelización de dicho algoritmo. Entre ellos la tesis de Leclerc [Leclerc92]. Con lo cual tiempo en la optimización aún se reduciría más.

12.6.7 Aplicaciones Reales para validar y contrastar los resultados teóricos y de simulación

Después de todo el trabajo presentado en esta tesis resulta evidente que para acabar de validar y contrastar los resultados teóricos y de simulación se deberá proceder a aplicar el nuevo algoritmo de generación de envolventes junto con el algoritmo de optimización asociado sobre aplicaciones reales. Recordemos que esta tesis nació de una aplicación real el proyecto ESPRIT TIGER, donde se pretendía detectar y diagnosticar fallos en tiempo real sobre un sistema complejo: una turbina de gas. Este proyecto en la actualidad se está continuando a través del proyecto TIGER II. La idea de la de poder aplicar este nuevo algoritmo en ese nuevo proyecto y además contrastar este nuevo algoritmo con ideas provenientes de otros grupos como el de Louise Travé del LAAS de Toulouse que están trabajando en el mismo proyecto utilizando sus propios enfoques de generación de envolventes.

12.6.8 Realización de Benchmarks con otras técnicas de Generación de Umbrales Adaptativos

Hemos visto que la generación de envolventes es una forma posible de generación de umbrales adaptativos pero existen otras formas. De esta diversidad de enfoques para resolver el problema de la generación de umbrales adaptativos ha nacido la idea de realizar una comparación entre dichos enfoques. Para ello se esta construyendo un ejemplo de sistema para poder realizar un benchmark este las diversas técnicas de generación de umbrales adaptativos. Inicialmente ya se está realizando un Benchmark entre nuestra técnica de generación de umbrales adaptativos basada en la generación de envolventes con las técnicas de generación de umbrales adaptativos propuestas por Frank y Ding [Ding94].

12.6.9 Generalización del nuevo algoritmo a sistemas con modelos no lineales

Finalmente, la idea es intentar generalizar este algoritmo de generación de envolventes para el caso de que se utilice un modelo no lineal del sistema. En principio, las ideas de la ventana temporal y de la optimización continúan siendo aplicables. Ahora se tratará de ver como intentar conseguir una cierta generalización en la obtención de la función objetivo, para poder tratar de una forma más o menos uniforme cualquier sistema con modelo no lineal.

Referencias

Referencias del Autor

R.1 Artículos y Publicaciones Realizadas a partir de la Tesis

"*Control Digital Directo utilizando las Toolboxes de MATLAB*" V. Puig, J.C. Hernández & J.Quevedo. I Congreso Nacional de Usuarios de MATLAB. Madrid 1995.

"*Sistema Informático para la Evaluación Atutomática de las Prestaciones de Controladores*". V. Puig & A. Escobet. XVI Jornadas de Automática. San Sebastián 1995.

"*Computer Tool for an Automatic Assessment of Industrial Controllers Perfomances*". V. Puig, A. Escobet & J. Quevedo. CONTROLO 1996. Congreso Nacional de Control Automático de Portugal. Oporto 1996.

"*Robust Fault Detection using Envelopes generated thorugh a parameter bounded model*". V. Puig, M. Brdys & J. Quevedo. TEMPUS Workshop in Advanced Control Systems. Viena 1996.

"*Applications in Fault Detection and Diagnosis of a new algorithm for Adaptive Threshold Generation*". V.Puig, J.Saludes & J.Quevedo. TEMPUS Workshop in Fault Diagnosis. Budapest 1997.

"*Determination of Window Length for a New Algorithm in Adaptive Threshold Generation*".J. Saludes, V. Puig & J. Quevedo. TEMPUS Workshop in Fault Diagnosis. Budapest 1997.

"*Un nuevo algoritmo para la Generación de Umbrales Adaptativos para la Detección y Diagnóstico Robusto de Fallos*". V. Puig, J. Saludes & J. Quevedo. XVIII Jornadas de Automática. Girona 1997.

"*Determinación de la Longitud Óptima de la Ventana Temporal para un nuevo algoritno de Generación de Umbrales Adaptativos*". J. Saludes, V.Puig & J.Quevedo. XVIII Jornadas de Automática. Girona 1997.

"*A new algorithm for Envelope Generation of a system with parameter bounded model: application to robust fault detection*". J.Saludes, V.Puig, M. Brdys & J.Quevedo. IEEE Transactions in Automatic Control. (*Enviado y en proceso de segunda revisión*).

"*Application of Global Optimization for Robust Fault Detection in Dynamical Systems*". V.Puig, J. Saludes & J.Quevedo. WESIC'98. Girona, Juny 1998.

"*Tolerance Generation for the Diagnostic Model Processor Algorithm (DMP)*". V.Puig, J.Saludes & J.Quevedo. 9th Symposium on Information Control in Manufacturing. INCOM'98. Nancy-Metz, Juny 1998.

"*A new algorithm for Envelope Generation of a System with Parameter Bounded Model using Optimization and the Sliding Window Principle*". V. Puig , J. Saludes & Joseba Quevedo. 7th International Conference in Information Processing and Management of Uncertainty in Knowledge-Based Systems. IPMU'98. Paris, Juliol 1998.

"*Application of Interval Constraint Satisfaction to a New Envelope Generation Algorithm*". V. Puig , J. Saludes & Joseba Quevedo. MISC'99. Girona, Febrer 1999. (*Enviado y en proceso de revisión*).

"*Overcoming Problems in Envelope Generation and in its Application to Fault Detection with a new Algorithm based on Global Optimization and the Sliding Window Paradigm*". J. Saludes, V. Puig & J. Quevedo. MISC'99. Girona, Febrer 1999. (*Enviado y en proceso de revisión*).

"*A New Algorithm for Adaptive Threshold Generation in Robust Fault Detection Based on a Sliding Window and Global Optimization*". V. Puig, J. Saludes & J. Quevedo. ECC'99. Germany, May 1999. (*Enviado y en proceso de revisión*).

R.2 Proyectos Fin de Carrera relacionados con la Tesis

"*Detecció Robusta de Falles fent servir Envolvents*". Alejando Sánchez. Curs 95-96. Escola d'Enginyers Industrials de Terrassa.

"*Sistema de Diagnosi de Falles fent servir un Sistema Expert*". Saturio Pascual. Curs 95-96. Escola d'Enginyers Industrials de Terrassa.

"*Sistema de Diagnosi de Falles d'una Planta Industrial fent servir l'algorisme DMP amb tolerància variable*". Guillem Sintes. Curs 96-97. Escola d'Enginyers Industrials de Terrassa.

"*Utilització de Tècniques Probabilístiques, de Lògica Fuzzy i d'Algebra d'Intervals per a la Generació d'Envolvents: aplicació a la detecció de falles en processos industrials*". Curs 96-97. Escola d'Enginyers Industrials de Terrassa.

"*Millora d'un Algorisme de Generació d'Envolvents fent servir Tècniques de Geometria Computacional*". Rubén Núñez. Curs 96-97. Escola d'Enginyers Industrials de Terrassa.

"*Generació d'Envolvents utilizant Tècniques d'Optimització Global*". Curs 96-97. José Maria Sánchez Escola d'Enginyers Industrials de Terrassa.

"*Implementació d'un Sistema de Generació d'Envolvents en Temps Real fent servir Programació Entera Mitxa*". Curs 97-98. José Lobato. Escola d'Enginyers Industrials de Terrassa i Universitat de Birmingham.

"*Estudi i aplicació de sistemes de Detecció i Diagnosi de Falles basats en models*". Josep Maria Vilardell. Curs 97-98. Escola d'Enginyers de Telecomunicació de Barcelona.

Bibliografía

Bibliografía

[Aguilar94] Aguilar-Martin, J., N. Rakoto-Ravalontsalama. "Uncertainty propagation in fuzzy simulation of dynamic systems. Application to a simplified turbine". Proceedings of the European Conference on Modelling and Simulation. Barcelona. pág. 519-522.

[Aguilar95] Aguilar-Martin J., P. Danes, R. Sarrate. (1995). "Estimation of trajectory envelops for fault detection". Modify Tempus SJEP's on Quantitative Fault Diagnosis. Hungary.

[Armengol98] Armengol, Q. (1998). "Simulation of dynamical systems with structured uncertainty: an overview". Research Report. Universitat de Girona.

[Armengol98b] Armengol, Q et al. (1998). "On Modal Interval Analysis for Envelope Determination within the Ca~En Qualitative Simulator". 7th International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems. La Sorbonne. Paris.

[Asaithambi82] Asaithambi, N, S. Zuhe, R, Moore. (1982). "On computing the range of values". Computing, 28, pág. 832-843.

[Basseville87] Basseville M, A. Benveniste, G. Moustakides, A. Rougée A. (1987) "Detection and diagnosis of changes in the eigenstructure of nonstationary multivariable systems". Automatica, Vol. 23, p. 479-489.

[Bazaraa93] Bazaraa, M.S., H.D. Sherali, C.M. Shetty. (1993). "Nonlinear Programming". John Wiley & Sons, 1993.

[Beard71] Beard R.V. (1971). "Failure in linear systems through self-organization". Rept. MTV-71-1, Man Vehicle Laboratory, MIT, Cambridge, MA.

[Beltran98] Beltran, M., Castillo, G., Kreinovich, V. (1998). "Algorithms That Still Produce a Solution (Maybe Not Optimal) Even When Interrupted: Shary's Idea Justified", Reliable Computing, 1998, Vol. 4, No. 1.

[Ben80] Ben-Haim Y. (1980). "An algorithm for failure location in a complex network". Nuclear Science and Engineering, 75, 191-199.

[Benhamou94] Benhamou, F., D. McAllester, P. Van Hentenryck. (1994). "CLP Intervals". Proceedings International Logic Programming Symposium. Ithaca.

[Benhamou97] Benhamou, F., Oler, W. (1997). "Applying interval arithmetic to real, integer and boolean constraints". The Journal of Logic Programming, pág 1-24.

[Berleant92] Berleant, D., B. Kuipers. (1992). "Qualitative-numeric simulation with Q3". Recent Adavances in Qualitative Physics. MIT Press. Cambridge. Massachussets.

[Bessiere93] Bessière, C., M. Cordier. (1993). "Arc-consistency arc-consistency again". Proceedings AAAI-93. Washington, DC. pág. 108-113.

[Blau69] Blau, G.E., Wilde, D.J. "Generalized polynomial programming". Canadian Journal on Chemical Engineering, 47, pág. 317-326.

[Bonarini94] Bonarini, A., G. Bontempi. (1994). "Qua Si: Qualitative Simulation approach for fuzzy models". Proceedings of the 1994 European Simultation Multiconference. Barcelona, pág. 420-424.

[Bonarini94b] Bonarini A., G. Bontempi (1994).. "A qualitative simulation approach for fuzzy dynamical models". ACM Transactions on Modeling and Computer Simulation (TOMACS) Vol. 4, Num. 4, 258-313.

[Bontempi96] Bontempi, G. (1996). "Qua Si III: a software tool for simulation of fuzzy dynamical systems". Proceedings of the European Simulation Multiconference (ESM96), Ghent, Belgium, pág 615-619.

[Bontempi97] Bontempi G. "Modeling with uncertainty in continuous dynamical systems: the probability and possibility approach". Report de Recherche. IRIDIA. Université Libre de Bruselles.

[Bousson94] Bousson K., L. Travé-Massuyès. (1994). "The CA-EN predictor: a Causal and Constrained-Based Qualitative Simulator". Proceedings of the Euroepean Conference on Modelling and Simulation". Barcelona. pág. 514-518.

[Bousson94] Bousson, K., L. Travé-Massuyés. (1994). "Putting more numbers in the qualitative simulator CA-EN". International Conference on Intelligent Systems Engineering. Hamburg, pág. 62-69.

[Borning77] Borning, A. (1977). "ThingLab: an object oriented system for building simulations using constraints". Proceedings IJCAI-77. Cambridge, pág. 497-499.

[Brooks81] Brooks, R.A. (1981). "Symbolic reasoning among 3-D models and 2-D images". Artificial Intelligence, 17, pág. 285-348.

[Chang93] Chang I-Chen, C. Yu, C. Liou. (1993). "Interval arithmetic approach to qualitative physics: static systems". Internation Journal of Intelligent Systems, vol. 8, pág. 405-430.

[Chow84] Chow E.Y., Willsky A.S. (1984) "Analytical redundancy and the design of robust failure detection systems". IEEE Transactions in Automatic Control; 29; 603-614.

[Clark89] Clark R.N. (1989). "State estimation for instrument fault detection". In Patton RJ, Frank PM, Clark RN (eds). "Fault diagnosis in dynamic systems, theory and application". Prentice Hall, Englewood Cliffs, NJ.

[Danes95] Danes P. (1995). "Interfaçage symbolique-numérique dans la simulation qualitative des systèmes dynamiques". PhD Disertation. LAAS-CNRS. Toulose.

[Danes95b] Danes P, J. Aguilar-Martin. (1995). "The symbolic-numeric interface: a "zosteric" approach". Rapport de Recherche. LAAS-CNRS. Toulose.

[Davis87] Davis, E. (1987). "Constraint propagation with inerval labels". Artificial Intelligence, 32 pág 281-331.

[Delmaire94] Delmaire G., J.P. Cassar, M. Staroswiecki. (1994). "Comparison of identification and parity space approach for failure detection in single-input single-output systems". IEEE Conference on control applications. Glasgow, vol 2, 865-870.

[Dechter90] Dechter, R. (1990). "Enhancement schemes for constraint processing: backjumping, learning and cutset descomposition". Artificial Intelligence, 41(3), pág. 273-312.

[Deville91] Deville, Y, P. Hentenryck. (1991). "An efficient arc consistency algorithm for a class of CSP problems". Proceedings of 12th International Joint Conference on Artificial Intelligence, Sydney, pág. 325-330.

[Ding89] Ding X., P.M. Frank. (1989) "Fault detection via optimally robust detection filters". 28th Conference on Decision and Control, Tampa, Vol. 2, 1767-1772.

[Ding91] Ding X., P.M. Frank PM. (1991) "Frequency domain approach and theshold selector for robust model-based fault detection and isolation". Proc IFAC/IMACS Symp. SAFEPROCESS '91, Baden-Baden, 307-312.

[Ding94] Ding X.,P.M. Frank. (1994). "Frequency domain approach to optimally robust residual generation and evaluation for model-based fault diagnosis". Automatica, Vol. 30, p.789-804.

[Duffin67] Duffin, R.J., E.L. Peterson, C.M. Zener. (1967). "Geometric Programming". John Wiley, New York.

[Duffin73] Duffin, R.J., E.L. Peterson. (1973). "Geometric programming with signomials". Journal on Optimization Theory Applications. 11, pág 3-35.

[Ecker80] Ecker, J.G. (1980). "Geometric programming: methods, computations and applications". SIAM Review, vol. 22, 3, pág 338-362.

[Emami88] Emami Naemi A, M.M. Akhter, S.M. Rock (1988). "Effect of model uncertainty on failure detection: the threshold selector". IEEE Transaction on Automatic Control, AC-33, p. 1106-1115.

[Falk69] Falk, J.E., R.M. Soland. (1969). "An algoritm for separable nonconvex programming problems". Mathematics of Operations Research, 1, 251-259.

[Falk73] Falk, J.E. (1973). "Global solutions of signomial problems". Tech. Rep. T-274. George Washington University. Program in Logistics, Washington DC.

[Fishwick91] Fishwick, P. (1991). "Fuzzy simulation: specifying and identifying qualitative models". Internation Journal on General Systems, vol. 19, pág. 295-316.

[Floudas92] Floudas, C.A., Pardalos, P.M. (1992). "Recent Advances in Global Optimization". Princenton Series in Computer Science. Pricenton University Press.

[Forbus84] Forbus, K.D. (1984). "Qualtitative Process Theory". Artificial Intelligence, 24, pág. 85-168.

[Frank89] Frank, P.M., J.Wunnenberg (1989). "Robust fault diagnosis using unknown input observers schemes". En "Fault Diagnosis in Dynamical Systems". Ed. R. Patton, P.Frank and R.Clark. Prentice Hall.

[Frank91] Frank P.M. (1991) "Enhancement of robustness in observer-based fault detection". IFAC Symposium SAFEPROCESS'91, Baden-Baden, Vol. 2, 275-287.

[Frank93] Frank, P.M. (1993) "Adavances in observer-based fault diagnosis". Proc. of the Conference TOOLDIAG '93, Toulouse, CERT, France; vol 3; 817-836.

[Fuente95] Fuente O'Connor, J.L. de La. (1995). "Tecnologías computacionales para sistemas de ecuaciones, optimización lineal y entera". Ed. Reverté.

[Gaganov81] Gaganov, A.A. (1981). "Computational complexity of the range of the polynomial in several variables". Leningrad University, Math. Department, M.S. Thesis.

[Gaganov85] Gaganov, A.A. (1985). "Computational complexity of the range on the polynomial in several variables". Cybernetics, 1985, pp. 418-421.

[Gao91] Gao Z, P. Antsaklis. (1991) "Stability of the pseudo-inverse method for reconfigurable control systems". International Journal of Control, Vol. 53, p. 717-729.

[Gardeñes86] Gardeñes, E.H., Mielgo, H., Trepat, A. (1986). "Modal Intervals: reasons and ground semantics". Lecture Notes in Computer Science, núm. 212. Interval Mathematics. Springer-Verlag.

[Ge88] Ge W., C.Z. Fang CZ. (1988) "Detection of faulty components via robust observation". International Journal of Control, Vol. 47, 581-599.

[Gertler85] Gertler, J., D. Singer (1985). "Augmented models for statistical fault isolation in complex dynamic systems". Proceedings of the American Control Conference, Boston, MA, 317-322.

[Gertler88] Gertler, J. (1988) "A survey of model-based failure detection and isolation in complex plants", IEEE Control Systems Magazine, 8, p. 3-11. 1988.

[Gertler90] Gertler, J., D. Singer (1990). "A new structural framework for parity equation based failure detection and isolation". Automatica, 26. 381-388.

[Gertler91] Gertler J (1991). "Analytical redundancy methods in fault detection and isolation". Proc. of the IFAC/IMACS Symposium SAFREPROCESS '91. Baden-Baden, vol 1, 9-21.

[Gertler93] Gertler J., M.M. Kunwer. (1993) "Optimal residual decoupling for robust fault diagnosis". International Conference on Fault Diagnosis, TOOLDIAG'93, CERT, p. 436-452.

[Gertler93b] Gertler J, R. Monajemy. (1993) "Generating directional residuals with dynamic parity equations". 12th IFAC World Congress, Sydney, Vol. 7, p. 505-510.

[Gill81] Gill, P.E., W. Murray, M.H. Wright. (1981). "Practical Optimization". Academic Press, London and New York.

[Grigor'ev88] Grigor'ev, D. Yu., Vorobjov. N.N. (1988). "Solving systems of polynomial inequalities in subexponential time". Journal of Symbolic Computation. Vol. 5, No. 1-2, pp. 37-64.

[Hansen92] Hansen, E. (1992) "Global Optimization using Interval Analysis". Marcel Dekker, New York.

[Horak88] Horak DT (1988). "Failure detection in dynamic systems with modelling errors" J. Guidance, Control and Dynamics, 11 (6), 508-516.

[Horst92] Horst, R., H. Tuy. (1992). "Global Optimization". Springer-Verlag.

[Horst95] Horst, R. , P.M. Pardalos (eds.), (1995). "Handbook of Global Optimization", Kluwer, Dordrecht.

[Hou91] Hou M., Muller P.C. (1991) "Design of robust observers for fault isolation". IFAC Symposium SAFEPROCESS'91. Vol. 1, 295-300.

[Hummel83] Hummel. R.A., Zucker, S.W. (1983). "On the fundations of relaxation labeling process". IEEE Transactions on Pattern Matching and Machine Intelligence, 5, pág. 267-287.

[Hyvönen92] Hyvönen, E. (1992). "Constraint reasoning based on interval arithmetic: the tolerance propagation". Artificial Intelligence, 58. pág. 71-112.

[Isaksson93] Isaksson, A.J. (1993) "An on-line threshold selector for failure detection". Proceedings of the International Conference TOOLDIAG'93. 628-634.

[Isermann91] Isermann R., B. Freyermuth. (1991) "Process fault diagnosis based on process model knowledge, Parts I (Principles for fault diagnosis with parameter estimation) and II (Case study experiments). ASME J. Dynamic Systems, Measurement Control; 113; 620-626, 627-633.

[Isermann93] Isermann R. (1993) "Fault diagnosis of machines via parameter estimation and knowledge processing". Automatica; 29; 815-836.

[Jones73] Jones H.L. (1973) "Failure detection in linear sytems". PhD thesis, Department of Aeronautics & Astronautics, MIT.

[Kay93] Kay, H., Kuipers, B. (1993). "Numerical behaviour envelopes for qualitative models". Proceedings of the Eleventh National Conference on Articial Intelligence (AAAI'93), 606-613.

[Kay95] Kay, H. (1995). "Semiquantitative simulation: successes, failures and future directions". Proceedings of the IJCAI-95. Workshop on Engineering Problems for Qualitative Reasoning. Montreal. Canada.

[Kearfott96] Kearfott, R.B. (1996). "Rigorous Global Search: Continuous Problems". Kluwer Academic Publishers.

[Kearfott96b] Kearfott, R.B., V. Kreinovich. (Eds.). (1996). "Applications of Interval Computations". Kluwer Academic Publishers.

[Kreinovich98] Kreinovich, V., Lakeyev, A. , Rohn, J., Kahl, P. (1998). "Computational Complexity and Feasibility of Data Processing and Interval Computations". Kluwe Academic Publishers.

[Kudva80] Kudva P, M. Widwanadham, A. Ramarkrishma. "Observers for linear systems with unknown inputs". IEEE Transaction on Automatic Control, AC-25, 113-115.

[Kuipers84] Kuipers, B.J. (1984). "Commonsense reasoning about causality: deriving behavior from structure". Artificial Intelligence, 24, pág. 169-203.

[Kuipers88] Kuipers, B.J., D. Berleant. (1988). "Using incomplete quantitative knowledge in qualitative reasoning". Proceedings 7th National Conference on Artificial Intelligence, St. Paul Minn, pág. 324-329.

[Kuipers94] Kuipers, B.J. (1994). "Qualitative Reasoning. Modelling and Simulation with incomplete knnowlege". MIT Press.

[Kurek83] Kurek, J.E. (1983) "The state vector reconstruction for linear systems with unknown inputs". IEEE Transaction on Automatic Control, AC-28, 1120-1122.

[Leclerc92] Leclerc, A.P. (1992). "Efficient and reliable global optimization", PhD Dissertation. Ohio State University.

[Lhomme93] Lhomme, O. (1993). "Consistency techniques for numeric CSPs". Proceedings IJCAI'93, pág. 232.238.

[Lou86] Lou, X. C., A.S.Willsky and G.C. Verghese (1986). "Optimally robust redundancy relations for failure detection in uncertain systems". IEEE Transactions on Automatic Control., AC-22, 333-344.

[MacFarlane76] Mac Farlane, A.G.J., Karcanias. (1976) "Poles and zeroes of linear multivariable systems: a survey of the algebraic, geometric and complex-variable theory". International Journal of Control, Vol. 24, Num. 1, p. 33-74.

[Mackworth77] Mackworth, K. (1977). "Consistency in networks of relations". Artificial Intelligence, 8(1), pág. 99-118.

[Marquez92] Marquez H.J., C.P.Diduch. (1992) "Sensitivity of failure detection using Generalized Observers". Automatica, Vol. 28, p. 837-840.

[Martínez98] Martínez Gasca, R. (1998). "Razonamiento y simulación en sistemas que integran conocimiento cualitativo y cuantitativo". PhD Dissertation. Universidad de Sevilla.

[Mohr86] Mohr, R., Thomas C. Henderson (1986). "Arc and Path consistency revisited". Artificial Intelligence, 28, pág. 225-233.

[Moore66] Moore, R.E. (1966). "Interval analysis". Prentice Hall.

[Moore79] Moore, R.E. (1979). "Methods and applications of interval analysis". SIAM Studies in Applied Mathematics. Philadelphia.

[Neitzke94] Neitzke, M., Neumann, B. (1994). "Comparative Simulation". Proceedings of 12th National Conference on Artificial Intelligence, Seattle, pág. 1205-1210.

[Neitzke94b] Neitzke, M., Neumann, B. (1994). "Simulating physical systems with relative descriptions of parameters". Proceedings of 11th National European Conference on Artificial Intelligence, Amsterdam, pág. 672-676.

[Ogryczak96] Ogryczak, W., Zorychta, K. (1996). "Modular optimizer for mixed integer programming MOMIP version 2.3". Working Paper WP-96-106. IIASA.

[Olin91] Olin PM, G. Rizzoni. (1991) "Design of robust detection filters". American Control Conference, p. 1522-1527.

[Passy67] Passy, U., Wilde, J. (1967). "Generalized polynomial optimizations". SIAM Journal on Applied Mathematics, 15, pág.1344-1356.

[Patton88] Patton R.J. (1988) "Robust Fault Detection using Eigenstructure Assignment". 12th IMACS World Congress Mathematical Modeling and Scientific Computation, Paris, Vol. 2, 431-434.

[Patton89] Patton R.J., P.M. Frank, R.N. Clarck. (1989) "Fault diagnosis in dynamic systems, theory and application". Prentice Hall, Englewood Cliffs, NJ.

[Patton94] Patton R.J., J. Chen. (1994). "Optimal selection of unknown input distribution matrix in the design of robust observers for fault diagnosis". Automatica, Vol. 29, Num. 4, p. 837-841.

[Petti92] Petti, Th. (1992) "Using mathematical models in knowledge-based control systems". PhD Dissertation, University of Delaware.

[Phatak88] Phatak M.S., N. Viswanadham (1988) "Actuator fault detection and isolation in linear systems". International Journal of Systems Science, Vol 19, 2593-2603.

[Pinter95] Pinter, J.D, (1995) "Global Optimization in Action". Kluwer, Dordrecht.

[Prosser93] Prosser, P. (1993). "Hybrid algorithms for the constraint satisfaction problems". Computational Intelligence, vol. 9, núm. 3, pág.268-299.

[Ratscheck84] Ratscheck, H., J. Rokne. (1984). "Computer Methods for the Range of Functions". Ellis Horwood, England.

[Ratscheck88] Ratscheck, H., J. Rokne. (1988). "New Computer Methods for Global Optimization". Ellis Horwood, England.

[Ratz94] Ratz, D. (1994). "Box-splitting strategies for the Interval Gauss-Seidel Step in a Global Optimization Method", Computing, vol. 53, pp. 337-354.

[SamHar95] Sam J. (1995). "Constraint consistency techniques for continuous domains". PhD Dissertation. Polytechnical School of Lausanne.

[Sauter93] Sauter D, G. Dubois, E. Levrat, J. Brémont. (1993) "Fault diagnosis in systems using fuzzy logic". EUFIT'93: First European Congress on Fuzzy and Intelligent Technologies", Aachen, 7-10 September 1993, Vol. 2, 781-788.

[Seliger91] Seliger R, P.M. Frank. (1991) "Fault diagnosis by disturbance decoupled nonlinear observers. 30th Conference on Decision and Control, Brighton, 2248-2253.

[Seliger93] Seliger R., P.M. Frank. (1993). "Robust residual evaluation by threshold selection and a performance index for nonlinear observer-based fault diagnosis". International Conference on Fault Diagnosis, TOOLDIAG'93, Toulouse, p. 496-504.

[Scheneider93] Scheneider H. (1993). "Implementation of a fuzzy concept for supervision and fault detection of robots". EUFIT'93: First European Congress on Fuzzy and Intelligent Technologies", Aachen, 7-10 September, Vol. 2, 775-780.

[Scheneider94] Scheneider H., P.M. Frank. (1994). "Fuzzy Logic based Threshold Selection for Fault Detection in Robots". 3rd IEEE Conference on Control Applications. Glasgow. 1994. 1127-1132.

[Shen92] Shen, Q., R. Leitch. (1992). "Combining qualitative simulation and fuzzy sets." In Recent advances in qualitative physics". MIT Press. Cambridge. Massachussets.

[Simmons86] Simmons, R. (1986). "Commonsense arithmetic reasoning". Proceedings AAAI-86. Philadelphia.

[Skelboe74] Skelboe, S. (1974). "Computation of rational interval functions". BIT, 14(1), pág. 87-95.

[Southwell40] Southwell, R. (1940). "Relaxation methods". Engineering Sciences, Clarendon, Oxford.

[Staroswiecki93] Staroswiecki M, J.P. Cassar, V.Cocquempot (1993) "Generation of optimal structured residuals in the parity space". Proc. IFAC 12th world congress, Sydney, vol 5, 535-542.

[Staroswiecki93b] Staroswiecki M, V. Cocquempot, J.P. Cassar. (1993) "A general approach for multicriteria optimization of structured residuals". International Conference on Fault Diagnosis, TOOLDIAG'93, CERT, p. 800-807.

[Stoer91] Stoer, J., R. Burlisch. (1991). "Introduction to numerical analysis". Volume 12 of "Texts in applied mathematics". Springer.

[Sussman80] Sussman, G.J., Steele, G.L. (1980). "CONSTRAINTS: a language for expressing almost hierarchical descriptions". Artificial Intelligence, 14, pág. 1-39.

[Sutherland63] Sutherland, I.E. (1963). "SKETCHPAD: a man-machine graphical communication system". MIT Lincoln Labs, Technical Report, 296. Cambridge.

[Trave96] Travé-Massuyes, L., R. Milne (1996). "Diagnosis of dynamical systems on explicit and implicit behauvioural models: an application to gas turbines". ESPRIT Project TIGER. Applied Artificial Intelligence Journal, 10/3.

[Trave97] Travé-Massuyes, L. R. Milne (1997). "TIGER: Gas Turbine condition monitoring using qualitative model based diagnosis". IEEE Expert Intelligent Systems and Applications, May-June.

[Vescovi91] Vescovi, M. (1991). "La Représentation des Connaissances et le Raisonnement sur les Systèmes Physiques". PhD Dissertation. University of Saboya.

[Vescovi92] Vescovi, M. (1992). "Numerical-based qualitative fuzzy simulation: an extension for piece-wise linear systems". Proceedings of Nono Congresso Brasileiro de Automatica.

[Vescovi95] Vescovi M., A. Farquhar, Y. Iwasaki. (1995). "Numerical interval simulation: combined qualitative and quantitative simulation to bound behaviours of non-monotonic systems". Proceeding of 14th International Joint Conference on Artificial Intelligence, pág. 1806-1812.

[Walker79] Walker B, E. Gai. (1979). "Fault detection threshold determination techniques using Markov theory". J. Guidance, Control & Dynamics, Vol. 1, Num. 8, p. 62-70. 1979.

[Walker89] Walker B. (1989). "Fault detection threhold determination using Markov theory". Chapter 14 in: [Patton RJ, Frank PM & Clark RN "Fault diagnosis in dynamic systems: theory and application" Prentice Hall.

[Watanabe82] Watanabe K., D.M. Himmelblau. (1982). "Instrument fault detection in systems with uncertainties". International Journal of Systems Scienc,. Vol. 13, 137-158.

[Weinmann91] Weinmann, A. (1991). "Uncertain Models and Robust Control". Springer.

[Weiss85] Weiss JL. "Threshold computation for detection of failures in SISO systems with transfer function errors". (1985). Proceedings of the American Control Conference. 2213-2218.

[Wilkinson92] Wilkinson, J.H. (1992). "The algebraic eigenvalue problem". Monographs on numerical analysis. Oxford University Press.

[Wünnenberg87] Wünnenberg J, P.M. Frank (1987). "Sensor fault detection via robust observers" en: "Sensor Fault Diagnostics, Reliability and Related Knowledge-Based Approaches". Tzafestas S, Singh M & Schmidt G (eds). Vol. 1, 147-160, Reidel Press.

[Wünnenberg90] Wünnenberg J. (1990) "Observer-based fault detection in dynamic systems". Ph.D. Thesis University of Duisburg, Germany.

[Zadeh65] Zadeh, L.A. (1965). "Fuzzy Sets". Information and Control. Num. 8, 338-353. 1965

[Zadeh78] Zadeh, L.A. (1978). "Fuzzy Sets as a basis for a theory of possibility". Fuzzy Sets and Systems. Vol 1. Num 1, 3-28.